Versione per
Commodore 64
Modulo Base + 2 Avventure









AVVENTURE

Guida pratica alla creazione di giochi di avventura

Enrico Colombini



GRUPPO EDITORIALE JACKSON

AVVENTURE

Guida pratica alla creazione di giochi di avventura

Enrico Colombini



GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Novembre 1985
GRAFICA E IMPAGINAZIONE: Roberto Pessina
COPERTINA: Sergio Mazzali
STAMPA: Stabilimento grafico A. Matarelli - Milano



Capitolo 1: non è mai troppo presto

Non è mai troppo presto per salvare i vostri preziosi programmi da un triste destino. Se saltate questo capitolo, non dite poi che non vi avevo avvertito.

La cassetta allegata alla confezione contiene le due avventure da giocare ed un programma per aiutarvi a scrivere le vostre avventure originali, il tutto ripetuto sul retro. Non correte il rischio di perdere o danneggiare i programmi: fate subito una copia di sicurezza (altrimenti detta backup).

Per fare la copia di backup, procuratevi tre cassette vergini, meglio se del tipo per computer (da pochi minuti, senza "coda" e con nastro adatto), accendete il calcolatore, riavvolgete la cassetta originale, scrivete:

LOAD "BASE"

e fate partire il registratore in PLAY. Quando il caricamento è terminato (il registratore si ferma e riappare il video), togliete la cassetta originale, inserite la prima delle tre cassette vergini, riavvolgetela, premete REC e PLAY e scrivete:

SAVE "BASE"

Quando ha finito, riavvolgete il nastro, scrivete:

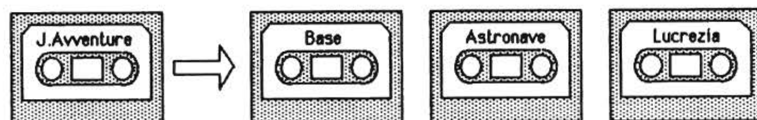
VERIFY

e fate partire il registratore in PLAY per verificare che la registrazione sia correttamente riuscita.
Ripetete la registrazione e la verifica di "BASE" anche sul retro della vostra cassetta, poi spezzate le due linguette di protezione poste sul retro, in modo da escludere cancellazioni accidentali.

Riprendete ora la cassetta originale e, senza riavvolgerla, ripetete altre due volte il procedimento con "ASTRONAVE" e "LUCREZIA", così:

LOAD "ASTRONAVE" (dall'originale)
SAVE "ASTRONAVE" (sulla seconda cassetta vergine, sui due lati)
VERIFY

E lo stesso per "LUCREZIA". Alla fine, avrete registrato tre cassette, la prima con "BASE", la seconda con "ASTRONAVE" e la terza con "LUCREZIA. Riavvolgete la cassetta originale e mettetela via in un luogo sicuro.



Se avete dei problemi a leggere i programmi originali, girate la cassetta e provate con il retro. Se non riuscite ancora, fate pulire, verificare e allineare il vostro registratore. Se proprio non c'è niente da fare, chiedete la sostituzione della cassetta.
Non usate più l'originale, che servirà solo da copia di sicurezza in caso di danneggiamento di una delle altre. cassette E' meglio usare cassette incise sul vostro registratore, in modo da non avere problemi di allineamento della testina. Inoltre, i tre programmi sono divisi e non rischiate di danneggiarli tutti insieme, ad esempio per uno stiramento del nastro.

Note:

- Se avete il disk drive, usate un dischetto inizializzato invece delle cassette ed aggiungete un ",8" quando fate il save: SAVE "BASE",8 (e così pure per gli altri due programmi).
- Sempre se avete il disk drive, probabilmente preferite salvare la situazione di gioco su disco, non su cassetta. In questo caso, modificate la linea 1095 prima di registrare i programmi, cambiando **di=0** in **di=1**.
- Se avete il "turbo tape", o un programma del genere, potete usarlo per registrare sulle vostre cassette. Tenete però presente che l'affidabilità, già non eccezionale in partenza, si riduce ulteriormente: è più facile avere problemi, specialmente se rileggete il programma con un registratore diverso da quello su cui è stato registrato.

Capitolo 2: all'Avventura!

Raccogliete le vostre cose e salutate gli amici: è giunta l'ora di partire per un mondo sconosciuto, dove vi aspettano gloria e fortuna. Oppure una fine solitaria ed ingloriosa. A voi la scelta!

Il fascino dei giochi di avventura sta nel loro potere: quello di creare un'illusione, facendo vivere il giocatore in un mondo immaginario, ma quanto mai reale per chi vi è immerso. Intendiamoci, non tutte le ciambelle riescono col buco: come ci sono romanzi avvincenti e polpettoni da quattro soldi, così si trovano avventure affascinanti ed altre semplicemente noiose. Riprenderò il discorso nei capitoli dedicati alla creazione delle vostre avventure.
L'avventura attrae anche per lo stesso motivo che rende interessanti rebus ed enigmi: la soddisfazione che si prova quando si riesce a risolverli, e che è tanto maggiore quanto più il problema era apparentemente insolubile. Anche qui, non basta che il problema sia difficile: deve essere alla portata del giocatore, altrimenti c'è solo frustrazione invece che divertimento.

Lo schema di un gioco di avventura è semplice: il calcolatore descrive (o disegna) il luogo in cui l'avventuriero si trova in quel momento. Il giocatore (che comanda l'avventuriero e si identifica con lui) può fare una serie di azioni, scrivendo semplicemente un comando alla tastiera. Per esempio:

APRI LA PORTA

e riceve in risposta la descrizione delle conseguenze. Ovviamente, la libertà di azione non è illimitata. Tutti i programmi di avventura "capiscono" almeno le direzioni, ad esempio:

NORD (o semplicemente N)

significa "spostati nel luogo adiacente a Nord di quello in cui ti trovi", e così pure per le altre direzioni (non dimenticando ALTO e BASSO).
Vagando per le stanze (o genericamente "luoghi"), capita di trovare degli

oggetti che possono essere utili. In questo caso, basta dire:

PRENDI IL PUGNALE

per appropriarsi dell'oggetto in questione, e:

LASCIA IL PUGNALE

per abbandonarlo nel luogo in cui ci si trova.
Un'altra azione di fondamentale importanza è GUARDA, che fornisce quasi sempre utili informazioni:

GUARDA IL FORZIERE

potrebbe dare come risposta: "E' aperto e contiene un tesoro" (anche se è molto improbabile che sia così facile...).
Se si incontrano altre persone, o comunque esseri più o meno intelligenti, può essere utile anche una conversazione:

PARLA CON IL DIAVOLO

che, magari, ha giusto un contratto da proporre (io non mi fiderei troppo...).
C'è sempre modo di conoscere gli oggetti che si possiedono in un dato momento, ad esempio scrivendo:

INVENTARIO

Di solito, è anche possibile registrare la situazione di gioco per riprenderla in un momento successivo, o per ripartire da un punto noto in caso di spiacevoli sorprese (ritrovarsi a cena con un drago, nel ruolo dell'antipasto, o cose del genere), con appositi comandi del tipo:

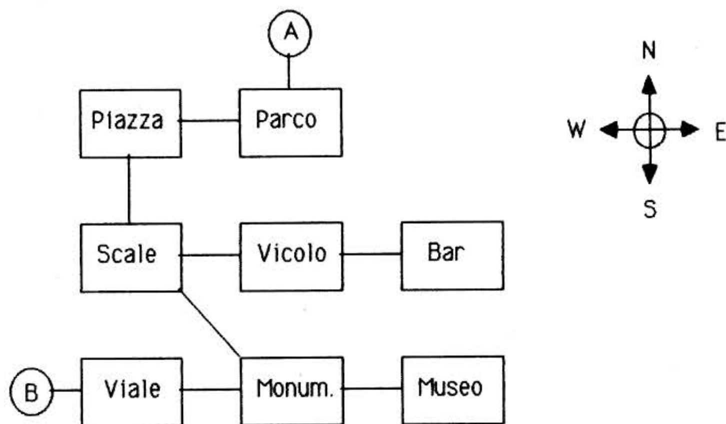
SALVA (per registrare) e
RIPRENDI (per ripartire dal punto dove avevate registrato)

Questi comandi, come dicevo, sono presenti in tutti i giochi di avventura (le parole non sono necessariamente le stesse). Le altre azioni ammesse sono lasciate alla fantasia dell'autore, ed alla qualità del programma. Le avventure meno interessanti sono quelle che ammettono una sola azione

per risolvere un certo problema, e rispondono "non capisco" in tutti gli altri casi. Le migliori sono ricche di risposte non direttamente utili alla soluzione del problema, ma che contengono prese in giro, indizi, o contributi al realismo dell'ambientazione.

Come si risolve un'avventura

La prima cosa da fare è tracciare una mappa quanto più possibile accurata, provando in ciascun luogo le sei possibili direzioni. Io uso una tecnica di questo genere: rappresento ogni luogo con un rettangolo, collegato ai luoghi adiacenti con una linea, così:



Dove non è possibile andare, si può sbarrare la direzione con un trattino, in modo da ricordarsi le direzioni provate, e non dimenticare di provarle tutte. Le linee diagonali, che partono dagli spigoli superiori od inferiori, indicano rispettivamente movimento verso l'alto e verso il basso (ad esempio: dalle scale si scende verso il monumento). Le lettere nei cerchiolini sono rinvii ad altri fogli dove prosegue la mappa, se non ci sta su un foglio solo.
E' fondamentale provare tutte le direzioni in tutti i luoghi, altrimenti si rischia di restare bloccati in situazioni senza uscita, quando magari ad Est

si stende un vasto territorio inesplorato. In casi disperati, può valer la pena di ricontrollare tutta la mappa alla ricerca di eventuali errori.

Una volta determinata la geografia dei luoghi (o almeno di quelli immediatamente raggiungibili), conviene osservare per bene tutti gli oggetti ed i dettagli che sembrano interessanti (con GUARDA). E' un'altra operazione essenziale, se non si vuole correre il rischio di perdere informazioni vitali.
Terminato questo lavoro preliminare, si può cominciare a raccogliere oggetti (prendendo nota sul solito foglio) ed a provare azioni di ogni genere. Ma attenzione: prima di comportarsi in modo sconsiderato, o comunque di compiere operazioni rischiose (come UCCIDI IL DRAGO), conviene registrare la situazione (con SALVA) per non dover ricominciare daccapo in caso di (probabile) esito disastroso.

Qualche altro consiglio, in ordine sparso:

- Se possibile, non giocate da soli. In compagnia ci si diverte molto di più ed è più facile avere buone idee.
- Leggete sempre le risposte con molta attenzione: spesso contengono indizi o informazioni essenziali, che possono sfuggire ad un lettore frettoloso.
- Provate tutto, ma proprio tutto quello che vi salta in mente, anche se vi sembra del tutto stupido, inutile o assurdo. Non è mai detto.
- Cercate di entrare nello spirito del gioco: alcune avventure richiedono forza bruta (ATTACCA IL DRAGO), altre giocano sull'astuzia o il ragionamento (PARLA CON IL DRAGO), altre sono surreali od originali (ACCAREZZA IL DRAGO).
- Tornate più volte negli stessi posti: la scena può essere cambiata (di solito c'è qualche indizio che lo suggerisce).
- Se una parola non viene accettata (ad esempio ROMPI), provate qualche sinonimo (come SPACCA, SPEZZA, FRANTUMA, DISTRUGGI) prima di concludere che l'azione non è quella giusta.

Ed ora, è giunto il momento di affrontare la prima avventura!

Capitolo 3: dallo spazio profondo al borgo medievale

Due avventure da giocare, la prima ambientata su un'astronave di linea che solo la vostra abilità può salvare da una sicura catastrofe, la seconda in pieno sedicesimo secolo, dove si dimostrerà se siete degni del titolo di cavaliere che portate.

Le due avventure contenute nella cassetta sono alquanto diverse tra loro, e non solo nell'ambientazione. L'astronave condannata è nata soprattutto per illustrare l'impiego del mio programma per la scrittura di semplici avventure in BASIC. La costruzione dell'avventura e del programma è descritta in dettaglio nel Quaderno Jackson "Come scrivere un'avventura". Anche se è di limitate dimensioni, si tratta comunque di un'avventura completa e ben giocabile. Gli avventurieri in erba vi troveranno la loro prima sfida, gli esperti possono usarla per scaldare i muscoli in vista della successiva (ma non è detto che anche loro la risolvano in dieci minuti!).

Per far partire il gioco, occorre innanzitutto inserire nel registratore la copia della cassetta contenente "ASTRONAVE". Se non avete fatto la copia come descritto nel Capitolo 1, è ora di farlo. Se infatti inserite l'originale, questa si auto-cancella (non è vero, ma ve lo meritereste...).
Per far partire il programma, scrivete:

LOAD "ASTRONAVE" (,8 se avete il disk drive)

e, quando ha finito di leggere dal registratore e torna il video:

RUN

Dopo qualche secondo, sarete a bordo (e nei guai).

Le azioni fondamentali sono quelle citate nel capitolo 2. Ve le riassumo, usando le maiuscole per evidenziare quello che voi scrivete (anche se in realtà lo scrivete in minuscolo):

N/S/E/O/A/B	sono le sei possibili direzioni.
PRENDI (qualcosa)	e
LASCIA (qualcosa)	servono per vari oggetti.
GUARDA (qualcosa)	fornisce utili indizi.
SALVA o SAVE	registra la situazione corrente (vedi nota).
RIPRENDI o LOAD	riprende la situazione registrata (vedi nota).
INVENTARIO, COSA o semplicemente '?'	elenca gli oggetti posseduti.

Il programma capisce varie altre parole, che vi appariranno (spero) ovvie nelle varie situazioni. Se proprio non ne venite fuori, potete ammettere la vostra incapacità e dare una sbirciatina alle linee di programma dalla 40000 in poi. Chissà che non vi suggeriscano qualcosa.

La seconda avventura, L'anello di Lucrezia Borgia, è molto più complessa e rifinita. Pur essendo ben lontana dal livello di dettaglio di "Avventura nel castello" (anche perchè, non facendo uso del disco, non c'è spazio per lunghe descrizioni), sono pronto a scommettere che darà filo da torcere anche agli avventurieri più smaliziati. Per farla partire, inserite la cassetta con "LUCREZIA" e fate:

LOAD "LUCREZIA" (,8 se avete il disk drive)

RUN

Dato che incontrerete varie persone, mi sembra il caso di rivolgere loro la parola: potrebbero darvi utili informazioni. Inoltre, ecco un prezioso consiglio: provate tutte le strade, e non solo in senso geografico. Non è detto che siano vicoli ciechi. Una volta capito lo spirito dell'avventura, le cose dovrebbero essere molto più facili. Se siete abbastanza svegli, naturalmente.

Nota:

Per registrare la situazione (SALVA) e rileggerla (RIPRENDI), usate un'altra cassetta vergine, altrimenti distruggete il programma (ma voi avete rotto le linguette di protezione, quindi non correte rischi, vero?).

Capitolo 4: creare un'avventura

Non è necessario essere esperti programmatori per scrivere un gioco di avventura: con l'aiuto del Modulo Base, basta una conoscenza elementare del BASIC.

Dopo aver giocato con "L'astronave condannata" e "L'anello di Lucrezia Borgia", sono sicuro che vi sarà venuta voglia di scrivere una vostra avventura. Come si fa? Da che parte si comincia? Aiuto!
Calma e gesso, come dicono i giocatori di bigliardo. Il lavoro si divide in due parti:

- Ideare la trama dell'avventura.
- Scrivere il programma che la realizza.

La prima è la parte creativa, la seconda richiede soltanto tecnica. E' un po' come dipingere un quadro: non basta avere l'immagine in mente, bisogna anche trasferirla sulla tela.
E' chiaro che le maggiori difficoltà vengono nella seconda parte: la scrittura del programma. Ci sono in circolazione, è vero, programmi già fatti che consentono di scrivere semplici avventure con poca fatica, ma obbligano l'autore a restare in uno schema predefinito. Inoltre, lasciano completamente all'oscuro del funzionamento interno del programma: funziona, e non chiedetevi perchè.
Ho preferito, invece, lavorare per due obbiettivi:

- Sollevare l'autore di avventure dal lavoro tecnico e noioso.
- Mostrare come funziona un gioco di questo tipo.

Ne è venuto fuori un programma che ho chiamato "Modulo Base", con il quale è possibile scrivere avventure con un minimo di conoscenza di BASIC, e con la massima libertà di introdurre varianti ed effetti speciali. Per mostrare che non c'è nulla di magico o misterioso in un programma del genere, l'ho scritto in BASIC "pulito", senza trucchi o routine in assembly.
Ovviamente, "L'astronave condannata" e "L'anello di Lucrezia Borgia" sono scritte impiegando il Modulo Base.

Il progetto ed il meccanismo di funzionamento del Modulo Base sono descritti in dettaglio nel Quaderno Jackson "Scrivere un'avventura". In queste pagine, vi guiderò invece nella realizzazione di una semplice avventura di esempio, per illustrarvi come si usa il programma.

Progettare un'avventura

Il progetto di un'avventura richiede prima di tutto un'ispirazione, cioè l'idea base della trama, l'ambientazione e lo schema generale della vicenda.
Una volta che si ha ben chiaro in mente quello che si intende fare, bisogna passare al dettaglio. Per poterla trasformare in un programma, la trama dev'essere accuratamente definita in ogni particolare:

- La mappa dei luoghi in cui si svolge l'avventura.
- Il dizionario, cioè l'elenco delle parole conosciute dal programma.
- Gli oggetti ed i personaggi che l'avventuriero può incontrare.
- Le azioni, cioè le frasi che hanno effetto sul gioco nelle varie situazioni.

Prima di iniziare a scrivere un'avventura, bisogna preparare una cassetta (o dischetto) di lavoro. Se usate un dischetto, formattatelo con OPEN 15,8,15,"N0:AVVENTURA,AV". In ogni caso, copiate il Modulo Base dalla vostra cassetta (o disco) su quella (quello) di lavoro:

```
LOAD "BASE"  (dalla cassetta o disco originale)
SAVE "BASE"  (sulla cassetta o disco di lavoro)
```

Alla fine, avrete una cassetta o dischetto che contiene solo il programma MODULO BASE. E' anche consigliabile preparare un'altra cassetta o dischetto inizializzato, da usare per le copie di sicurezza del lavoro in corso.

Capitolo 5: Operazione Zanna Bianca

Prima di cominciare a lavorare sul programma BASIC, bisogna impostare, almeno nelle linee generali, la trama dell'avventura che si vuol realizzare.

Non commettete anche voi l'errore di mettervi a scrivere un'avventura senza aver ben chiara in mente la trama ed i dettagli. Risparmiereste, sì, un poco di lavoro all'inizio, ma finireste col pagarlo caro. Andando avanti nella scrittura del gioco, finireste col trovarvi persi in una miriade di dettagli che non collimano, come in un giallo scritto male. Meglio, dunque, perdere qualche ora per progettare l'avventura nei minimi particolari.
Tanto per smentirmi subito, non seguirò il mio stesso consiglio, e costruirò un esempio un passo alla volta, aggiungendo man mano nuovi dettagli. Sia chiaro che lo faccio soltanto per non presentarvi le difficoltà tutte insieme. Inoltre, l'esempio è veramente molto semplice. E poi, me lo sono preparato nei dettagli <u>prima</u> di iniziare a scrivere.

La micro-avventura che userò per illustrarvi come si lavora col Modulo Base si chiama "Operazione Zanna Bianca". E' la vicenda di un agente segreto, che deve "prelevare" con destrezza dei microfilm riguardanti le rotte polari dei sommergibili nucleari (non si dice di quale Potenza: sceglietе a vostro piacimento).
E' chiaro che non si tratta di una vera avventura, ma solo di un esempio che, tuttavia, contiene quasi tutti gli elementi usabili per la costruzione di avventure vere e proprie. Non scandalizzatevi, dunque, per la semplicità e la mancanza di dettaglio. Anzi, se volete poi trasformare l'esempio in una vera avventura, la strada è aperta per la vostra fantasia.

Prima di passare alla parte tecnica, due parole sulla creazione delle avventure. Ci sono alcuni punti da tenere ben presenti, se non volete semplicemente aggiungere un'altra avventura insulsa alle troppe che già circolano. Cercherò di indicarvi i principali:

- La trama dev'essere coerente. Le incoerenze spezzano il filo della vicenda, e rompono il fragile incantesimo che tiene avvinto il

lettore/giocatore (ci sono molti punti in comune tra un'avventura ed un racconto). Parole magiche non hanno senso a bordo di un'astronave, come un tesoro stona in un'avventura di spionaggio: fa cadere la tensione e distrae il giocatore dallo scopo principale.

- L'atmosfera è importante, per fare in modo che il giocatore si immedesimi col personaggio. Anche se il sistema che vi propongo non permette lunghe descrizioni, basta spesso un aggettivo per stimolare la fantasia del giocatore. "Una sala" è semplicemente una sala, ma "Una sala silenziosa" è tutt'altra cosa: chi immagina la polvere dei secoli depositata sugli antichi arredi, chi sospetta che qualcosa si celi dietro il silenzio innaturale... In ogni caso, non è un posto qualunque (anche se, magari, non c'è proprio niente di utile).

- Lo scopo dell'avventura è il divertimento dei giocatori, non il sadico compiacimento dell'autore che nessuno riesca a risolverla. I problemi devono essere abbastanza difficili da dare soddisfazione quando li si risolve, ma non impossibili: altrimenti nessuno si diverte. Bisogna collaudare l'avventura, cioè farla giocare a vari amici: se troppo pochi riescono a risolvere un certo problema, è il caso di fornire qualche indizio in più.

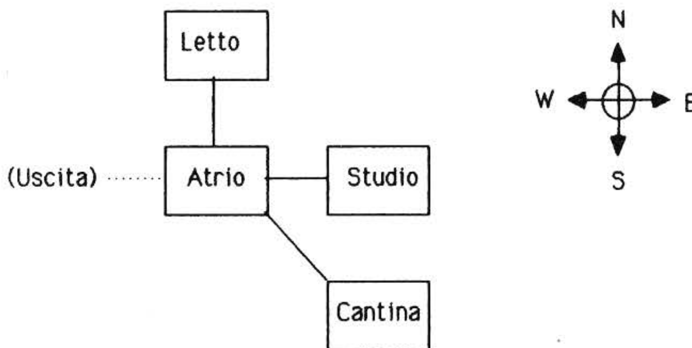
- Evitare l'errore più grave: che le difficoltà siano risolubili con una sola azione, descritta con una sola frase valida. I collaudi servono soprattutto a questo. Se avete previsto l'azione "Rispondi al telefono", troverete che molti scrivono invece "Prendi il telefono" o "Alza la cornetta" o simili. Se ottengono solo risposte del tipo "Non capisco" o, peggio, "Non succede niente", probabilmente dopo un po' lasciano perdere.

- Sono i dettagli che rendono interessante un'avventura, molto più che la complessità della vicenda. Più risposte inutili o secondarie sono state previste, più il gioco è appassionante ed i giocatori si divertono. Per esempio, in risposta ad "Accendi il televisore", una risposta tipo "Stanno trasmettendo la 257ª puntata di Dallas" è senz'altro meglio del solito "Non capisco". Anche qui, i collaudi sono molto importanti: prendete nota di tutto quello che scrivono i vostri amici, ed aggiungete risposte adeguate. I giocatori non riusciranno a staccarsi dal computer, e si chiederanno come diavolo avete fatto a prevedere tutte quelle possibilità.

Capitolo 6: la casa dell'ambasciatore

Il programma si occupa automaticamente degli spostamenti. Tutto quello che c'è da fare è introdurre le descrizioni dei luoghi ed i loro collegamenti nelle apposite linee di DATA del Modulo Base.

Ovviamente, i microfilm si trovano nell'appartamento dell'ambasciatore. Per semplicità, tralascio tutte le stanze accessorie (bagno, cucina, ecc.), e considero soltanto quelle che hanno a che fare con l'avventura. Questa è la mappa stilizzata dell'appartamento:



Come si vede, dall'atrio si può andare a Nord (camera da letto), ad Est (Studio) ed in basso (cantina). Si potrà anche andare ad Ovest per uscire dal gioco, ma per ora non ci interessa (non esiste, in effetti, un altro luogo ad Ovest).
Elenchiamo per esteso i vari luoghi possibili, numerandoli a partire da uno ed indicando i vari collegamenti:

Luogo:	Nord	Sud	Est	Ovest	Alto	Basso
1) Camera da letto	-	atrio	-	-	-	-
2) Atrio	letto	-	studio	-	-	cantina
3) Studio	-	-	-	atrio	-	-
4) Cantina	-	-	-	-	atrio	-

I numeri dei luoghi sono importanti: nel programma sono l'unico modo per distinguere un luogo dall'altro. La camera da letto è il numero 1, l'atrio è il numero 2, e così via. Anzi, vale la pena di riscrivere la tabella mettendo in evidenza i numeri (anzichè i nomi) dei vari luoghi:

Luogo:	Nord	Sud	Est	Ovest	Alto	Basso
1) Camera da letto	-	2	-	-	-	-
2) Atrio	1	-	3	-	-	4
3) Studio	-	-	-	2	-	-
4) Cantina	-	-	-	-	2	-

Questa tabella rappresenta la mappa della nostra avventura. Ci sono infatti tutte le informazioni necessarie per determinare l'effetto di uno spostamento. Ad esempio, muovendosi in alto dal luogo 4 (cantina) si finisce nel luogo 2 (atrio), mentre uno spostamento in basso dal medesimo luogo 4 (cantina) non è possibile. Ora che la tabella è pronta, non resta che introdurla nel Modulo Base.

Introdurre la mappa

Per lavorare sul Modulo Base, occorre caricarlo. Dunque, fate:

LOAD "BASE" (,8 se avete il disk drive, d'ora in poi è sottinteso)

Dato che l'avventura contiene messaggi da stampare, è meglio fare in modo che il video del C64 possa mostrare maiuscole e minuscole. Se non l'avete

già fatto, premete inieme SHIFT e C= in modo da vedere le minuscole.

E' il caso di cambiare subito le prime tre linee di REM:

```
100 rem ** avventura:  interprete  e modulo  base **
110 rem **          di enrico colombini          **
120 rem ** (c)1985 dinosoft  e jackson editrice  **
```

sostituendole con qualcosa del genere:

```
100 rem ** avventura:  operazione  zanna bianca **
110 rem **            di pinco pallino          **
120 rem **       diffidate delle imitazioni!    **
```

Compiuta questa importante formalità, possiamo passare a qualcosa di più sostanzioso. Osservate le linee:

```
44970 rem
44980 rem - mappa:
44990 rem
45000 data "fm": rem --- luoghi e mappa, dal luogo n.1 ---
```

E' proprio qui, a partire dalla linea 45000, che vanno inserite le caratteristiche dei luoghi, in ordine di numero. Il primo elencato sarà il numero 1, il secondo il numero 2, e così via. Iniziamo dunque dal numero 1:

```
45000 data "in un'ampia camera da letto","000200000000"
```

Cosa significa tutto questo? Procediamo con ordine. Ogni linea deve iniziare con il numero di linea, seguito dall'istruzione DATA (nel testo, uso il maiuscolo). Per comodità, conviene usare numeri di linea di 10 in 10. La prima linea di DATA della mappa deve essere la numero 45000.
Poi va messa la descrizione del luogo (tra virgolette), considerando che verrà stampata preceduta da "Sei " e seguita da un punto. Il luogo numero 1 appena introdotto verrà dunque stampato così:

```
Sei in un'ampia camera da letto.
```

Infine, va messo l'elenco dei luoghi adiacenti. Ricordate la tabella dei collegamenti? Questa era la linea riguardante la camera da letto:

1) Camera da letto - 2 - - - -

Riscriviamola usando solo numeri di due cifre, dunque 02 invece di 2, ed indicando con 00 le direzioni non ammesse (dove non si può andare):

1) Camera da letto 00 02 00 00 00 00

Scriviamo ora di seguito i sei gruppi di due cifre (riguardanti nell'ordine: Nord, Sud, Est, Ovest, Alto, Basso), ottenendo una stringa di 12 cifre:

000200000000

Questa stringa va riportata (tra virgolette) dopo la descrizione del luogo, separata da questa con una virgola, ottenendo per l'appunto:

```
45000 data "in un'ampia camera da letto","000200000000"
```

A stretto rigore, le virgolette non sono obbligatorie. Se non le mettete, fate però attenzione: non potete introdurre caratteri come virgola, due punti e punto e virgola nella descrizione del luogo, ed eventuali spazi in più possono creare problemi. A scanso di guai, è meglio metterle.
La lunghezza massima della descrizione è limitata dalla lunghezza di una linea di programma. Diciamo che potete arrivare a poco più di 50 caratteri, a causa della limitazione ad 80 caratteri di una linea del Commodore (volendo, potete mettere la stringa di 12 cifre in una linea di DATA a sè, guadagnando quindici caratteri ed arrivando quasi a 70).
Sistemato il primo luogo, ripetiamo il medesimo procedimento per i successivi. Alla fine, questo è il risultato:

```
45000 data "in un'ampia camera da letto","000200000000"
45010 data "in un atrio arredato con stile","010003000004"
45020 data "nello studio dell'Ambasciatore","000000020000"
45030 data "in una cantina ammuffita","000000000200"
45040 data "fm"
```

Queste linee comunicano al programma tutto quanto è necessario sapere



sui luoghi dove si svolge l'avventura e la loro disposizione. Fate molta attenzione a non dimenticare la linea finale:

```
45040 data "fm"
```

altrimenti il programma non sa quando i luoghi sono finiti e si ferma con errore prima ancora di partire (indicando un errore intorno alla linea 1100), o comunque si confonde parecchio.
A questo punto, è il caso di salvare il programma prima di provarlo, sull'altro lato della cassetta:

SAVE "ZANNA1" e poi VERIFY

Girando la cassetta ad ogni SAVE, si hanno sempre le ultime due versioni.

Verifica e collaudo della mappa:

Per controllare il lavoro fatto fino a questo momento, fate:

RUN

Apparirà la scritta "**Un attimo di pazienza...** " (mentre il programma si legge i DATA), poi lo schermo verrà cancellato ed apparirà "- **Premi <spazio> per continuare** -". Se invece vi capita un **?SYNTAX ERROR**, un **?OUT OF DATA ERROR** o qualcosa del genere, significa che avete sbagliato ad introdurre i luoghi: verificate bene che ogni linea inizi con DATA (minuscolo nel programma), seguito dalla descrizione del luogo (tra virgolette), poi una virgola e le 12 cifre dei collegamenti (sempre tra virgolette, e controllate che siano proprio 12); controllate infine che ci sia "fm" alla fine (minuscolo e tra virgolette).

Se va tutto bene, obbedite al programma e premete la barra spaziatrice. Leggerete sul video:

Sei in un'ampia camera da letto.

Cosa devo fare?

Già, cosa dovete fare? Per ora, dovete provare a spostarvi nelle varie direzioni, per verificare che la mappa sia corretta. Rispondete dunque:

Cosa devo fare? nord

(o semplicemente **n**, comunque minuscolo). Il programma risponde:

- Di li' non puoi andare.

e questo corrisponde con la nostra mappa. Scrivendo invece:

Cosa devo fare? sud

succede che:

Sei in un atrio arredato con stile.
Cosa devo fare?

Ora, c'è da fare un lavoro noioso ma necessario: provare tutte le sei possibili direzioni (N/S/E/O/A/B) in tutti i luoghi, per verificare che la mappa sia corretta in ogni particolare. Se trovate qualche errore, correggetelo nella corrispondente linea di DATA.
In alcune avventure, può darsi che vi siano luoghi non direttamente collegati tra loro: ad esempio, luoghi in cui si arriva prendendo un aereo, o trasportati da una parola magica. Come fare a controllare anche questi, se non ci si può arrivare? Vi insegno un trucco.
Supponiamo che lo studio non sia direttamente raggiungibile dall'atrio, o comunque dal luogo in cui vi trovate. Per andarci, quando il programma scrive:

Cosa devo fare?

tenete premuto il tasto RUN/STOP e date un rapido colpetto al tasto RESTORE. Il programma, colpito alla schiena, si arresterà bruscamente rimettendo il video nel colore originario (e cancellandolo). Rimettetelo subito in modo minuscole con SHIFT-C=

Ora il programma è fermo. Se lo desiderate, potete anche listare (LIST) una

o più linee, ma fate attenzione a non modificare alcuna linea, altrimenti perdete tutte le variabili, cioè le informazioni che il programma si era letto dai DATA o creato per i fatti suoi.
Per andare nello studio, che è il luogo 3 (terza linea dei DATA della mappa), scrivete:

lu=3:goto 760 (la prima lettera è una elle minuscola)

Otterrete:

Sei nello studio dell'ambasciatore.

Cosa devo fare?

Siete arrivati direttamente alla vostra destinazione: il luogo 3, cioè lo studio. Questo sistema, come vedremo, servirà molto nella messa a punto del gioco.
Tecnicamente, avete fermato il programma (RUN/STOP-RESTORE), cambiato il valore di una variabile (lu=3), e fatto ripartire il programma dall'inizio del ciclo di gioco (goto 760). Per gli esperti: non è possibile usare l'istruzione CONT per far ripartire il programma, perchè non funziona durante una INPUT, e perchè non è stato fermato con RUN/STOP (che, comunque, ha lo stesso problema).
Avete imparato che la variabile **lu** contiene il numero del **luogo corrente**, cioè quello in cui si trova l'avventuriero. Cambiando numero, si cambia luogo.
A proposito, quando volete fermare il programma per lavorarci sopra, fate RUN/STOP-RESTORE come prima (non dimenticando poi SHIFT-C=).

Introduzione all'avventura

Avrete notato, nel collaudo, che il gioco inizia in camera da letto. Perchè proprio lì? Perchè la camera da letto è il luogo numero 1 (il primo nell'elenco dei DATA). Non è però obbligatorio partire dal luogo 1, anzi è possibile far iniziare l'avventura nel luogo preferito. Nel nostro caso, direi che il luogo più indicato è l'atrio, cioè il luogo numero 2.
C'è anche un'altra cosa da fare: scrivere una breve presentazione ed

introduzione all'avventura, che viene stampata ogni volta che si inizia il gioco.
Entrambe queste cose sono compito di un'apposita routine di introduzione al gioco, che si trova alla linea 4500. Al momento, è alquanto scarna:

```
4480 rem - introduzione -
4490 rem
4500 print chr$(147):print:print:print
4510 rem --- mettere qui introduzione avventura ---
4520 rem --- ricordarsi lu=luogo iniziale ---
4950 lu=1:print:return
```

Notate che la linea 4950 contiene **lu=1**. E' questo che fa in modo che il gioco inizi nel luogo 1. Basta una piccola modifica:

```
4950 lu=2:print:return
```

E, facendo RUN:

dopo il solito "- Premi <spazio> per continuare -", il gioco inizia con:

Sei in un atrio arredato con stile.

Risolto il problema di far iniziare il gioco nel luogo voluto, possiamo scrivere l'introduzione. Si tratta soltanto di mettere alcune istruzioni PRINT, a partire dalla linea 4510:

```
4510 print "*** Operazione Zanna Bianca ***"
4520 print:print
4530 print "Questa e' la tua missione:"
4540 print "Recuperare i microfilm o morire!"
```

Quello che va stampato dev'essere tra virgolette, e preceduto dall'istruzione PRINT. Le due PRINT a vuoto alla linea 4520 servono per stampare due linee vuote di stacco. Nel complesso, la routine di introduzione deve presentarsi così:

```
4480 rem - introduzione -
4490 rem
```

```
4500 print chr$(147):print:print:print
4510 print "*** Operazione Zanna Bianca ***"
4520 print:print
4530 print "Questa e' la tua missione:"
4540 print "Recuperare i microfilm o morire!"
4950 lu=2:print:return
```

Gli esperti potranno notare che ci starebbe tutto in meno linee, separando le varie istruzioni con due punti. Ho preferito la chiarezza, anche se occupa un poco di spazio in più.
Ora, salvate il programma, non dimenticando di girare la cassetta:

SAVE "ZANNA2" e VERIFY

Perchè "ZANNA2" e non "ZANNA1"? Perchè, se avete il disco, non conviene salvare sempre con lo stesso nome, altrimenti un qualunque problema (ad esempio una mancanza di corrente, o lo sportellino del drive chiuso male) causa la perdita di tutto il lavoro precedente. Inoltre, è possibile risalire alle versioni precedenti in caso di modifiche errate. Se avete la cassetta, non vi confondete su qual'è l'ultima versione
Ed ora, il solito RUN di prova:

Un attimo di pazienza...

e poi:

***** Operazione Zanna Bianca *****

Questa è la tua missione:
Recuperare i microfilm o morire!

- Premi <spazio> per continuare -

Naturalmente, ci vorrebbe un'introduzione più sostanziosa, ma questo è solo un esempio. Potete scrivere tutto quello che vi pare.
Il "- Premi <spazio> per continuare -" alla fine è automatico. Se volete

introdurne altri, ad esempio perchè l'introduzione non ci sta in un solo video, usate l'istruzione:

```
gosub 1500
```

per fermare la stampa tra una PRINT e la successiva. Non mettete pause (ritardi) nell'introduzione, sono molto fastidiose (anzi, per ora non vi spiego nemmeno come si fa).

E' tutto chiaro?

Spero di sì. Non vi conviene andare avanti senza aver assimilato, e provato in pratica, il contenuto di questo capitolo.
Un riassunto molto sintetico di quello che ho finora illustrato, utile per non dover cercare informazioni diluite in varie pagine, è riportato nell'Appendice B.


DAF

Capitolo 7: chiave, porta e microfilm

In un'avventura, si incontrano oggetti di vario tipo. Per fare in modo che siano trattati dal programma in modo automatico, basta introdurre il loro nome nel dizionario e le loro caratteristiche in un apposito elenco.

Per avere un'avventura, non basta una mappa. Bisogna anche che ci siano dei problemi da risolvere. Di solito, si tratta di azioni da compiere su oggetti, per esempio:

APRI LA PORTA

Perchè questo sia possibile, occorre che:

- Il programma "capisca" le parole APRI e PORTA.
- Ci sia un oggetto PORTA su cui agire.
- L'azione APRI LA PORTA sia stata prevista.

Partiamo dal primo punto: se vogliamo che il programma riconosca le parole che scriviamo, bisogna che in qualche modo le conosca. Prima ancora, bisogna che noi ci prepariamo un elenco delle parole che potranno servire nel corso dell'avventura. Anche se è possibile aggiungerle una alla volta, è senz'altro più comodo pensarne quante più possibile ed introdurle in un colpo solo.
Per decidere quali parole insegnare al programma, dobbiamo avere un'idea di quello che può succedere durante il gioco. Per non complicare troppo le cose, schematizziamo l'avventura in questo modo:

- I microfilm sono nello studio.
- La porta che dall'atrio dà nello studio è chiusa a chiave.
- La chiave si trova in cantina.

Ora, scriviamo la sequenza principale, cioè la serie di azioni che porta alla soluzione del gioco, a partire dall'atrio:

B (o BASSO)	(va in cantina)
PRENDI LA CHIAVE	(nessuna difficoltà)
A (o ALTO)	(va nell'atrio)
APRI LA PORTA	(solo se ha la chiave)
E (o EST)	(va nello studio, solo se la porta è aperta)
PRENDI I MICROFILM	(nessuna difficoltà)

Manca un finale, ma ci penseremo dopo. Queste sono dunque le parole che il programma deve per forza conoscere, perchè si possa arrivare in fondo all'avventura:

APRI, PRENDI	(verbi)
CHIAVE, PORTA, MICROFILM	(oggetti)

Le parole che il programma conosce sono raccolte in un apposito **dizionario**, nel quale ci sono già alcuni vocaboli fondamentali:

```
39970 rem
39980 rem - dizionario:
39990 rem
40000 data "?",13,"a",5,"agli",7,"al",7,"all",7
40010 data "alla",7,"alle",7,"allo",7,"alto",5
40020 data "b",6,"basso",6
40030 data "col",7,"con",7,"cosa",13
40040 data "e",3,"est",3
40050 data "gli",7,"guarda",10
40060 data "i",7,"il",7,"inventario",13
40070 data "l",7,"la",7,"lascia",9
40080 data "le",7,"lo",7,"load",12
40090 data "n",1,"nord",1
40100 data "o",4,"ovest",4
40110 data "posa",9,"prendi",8
40120 data "riprendi",12
40130 data "s",2,"sali",5,"salva",11
40140 data "save",11,"scendi",6,"sud",2
40150 data "un",7,"una",7,"uno",7
40160 data "w",4
40170 data "fd": rem --- ricordarsi l'ordine alfabetico!!! ---
```

Notiamo che il vocabolario contiene già:

- Le sei direzioni (infatti, il programma le capisce).
- Articoli e preposizioni (che vengono semplicemente ignorati).
- Il verbo GUARDA.
- I verbi PRENDI e LASCIA (o POSA).
- I comandi SALVA (o SAVE) e RIPRENDI (o LOAD).
- Il comando COSA (o INVENTARIO, o '?').

Dunque, PRENDI è già contenuto nel dizionario. Ci restano da inserire APRI, CHIAVE, PORTA e MICROFILM. Per farlo, occorre assegnare un **codice** a ciascuna parola.
Mi spiego: osservando il dizionario, potete vedere che ogni parola è seguita da un numero: il suo codice, appunto. Per esempio, LASCIA (linea 40070) è seguita dal numero 9: vuol dire che 9 è il codice di LASCIA. In tutto il resto del programma, non verrà mai usata la parola LASCIA, ma il suo codice (9). Questo permette un notevole risparmio di tempo e memoria, ed inoltre rende semplice introdurre sinonimi, cioè parole con lo stesso significato. Ad esempio, nella linea 40110 vedete che anche la parola POSA ha codice 9. Per dire che due parole sono equivalenti, basta dare il medesimo codice.

Dobbiamo dunque assegnare un codice a ciascuna delle nostre parole. Non c'è motivo particolare di usare un numero piuttosto che un altro, ma ci sono queste limitazioni:

- I codici ammessi vanno da 1 a 98.
- I codici da 1 a 13 sono già usati.

Inoltre, io ho trovato comodo (ma non è per niente obbligatorio) raggruppare i vocaboli dello stesso tipo. Ad esempio, in "L'anello di Lucrezia Borgia" ho usato i codici da 20 in poi per i verbi, quelli da 40 in poi per gli oggetti prendibili (come l'anello) e quelli da 60 in poi per gli oggetti non prendibili, o i personaggi (come la strega). Usando lo stesso sistema, darei questi codici:

APRI	20
CHIAVE	40
MICROFILM	41
PORTA	60

Stabiliti i codici, non resta che introdurre in dizionario le nuove parole. C'è una regola fondamentale da rispettare: l'ordine alfabetico. Cominciamo con l,inserire APRI al suo posto, tra virgolette e seguito dal suo codice (20):

```
40010 data "alla",7,"alle",7,"allo",7,"alto",5,   "apri",20
40020 data "b",6,"basso",6
```

Dato che APRI sta (in ordine alfabetico) tra ALTO e B, va inserito in mezzo tra questi due vocaboli. E' la stessa cosa separarlo dal precedente con una virgola o metterlo in una linea tutta per sè:

```
40010 data "alla",7,"alle",7,"allo",7,"alto",5
40015 data "apri",20
40020 data "b",6,"basso",6
```

Quest'ultima soluzione è più comoda, la precedente era più compatta. Dato che l'avventura è molto piccola e mi interessa la chiarezza, userò la seconda. Occorre fare attenzione a non mettere virgole in fondo alle linee. Le virgole servono solo per separare i dati tra loro, come se fossero su linee distinte. Le parole vanno in minuscolo.
Se avessimo dovuto inserire, che so, ALLEATO, sarebbe stato necessario modificare la linea 40010 (non fatelo, è solo un esempio):

```
40010 data "alla",7,"alle",7, "alleato",66, "allo",7,"alto",5
```

o spezzarla:

```
40010 data "alla",7,"alle",7, "alleato",66
40012 data "allo",7,"alto",5
```

Una volta capito il meccanismo, è molto semplice. Per modificare il programma, assicuratevi di averlo in memoria, o fate:

LOAD "ZANNA2"

ed inserite le nuove parole nel dizionario, che alla fine deve presentarsi in questo modo:

```
39970 rem
39980 rem - dizionario:
39990 rem
40000 data "?",13,"a",5,"agli",7,"al",7,"all",7
40010 data "alla",7,"alle",7,"allo",7,"alto",5
40015 data "apri",20
40020 data "b",6,"basso",6
40025 data "chiave",40
40030 data "col",7,"con",7,"cosa",13
40040 data "e",3,"est",3
40050 data "gli",7,"guarda",10
40060 data "i",7,"il",7,"inventario",13
40070 data "l",7,"la",7,"lascia",9
40080 data "le",7,"lo",7,"load",12
40085 data "microfilm",41
40090 data "n",1,"nord",1
40100 data "o",4,"ovest",4
40105 data "porta",60
40110 data "posa",9,"prendi",8
40120 data "riprendi",12
40130 data "s",2,"sali",5,"salva",11
40140 data "save",11,"scendi",6,"sud",2
40150 data "un",7,"una",7,"uno",7
40160 data "w",4
40170 data "fd": rem --- ricordarsi  l'ordine  alfabetico!!!  ---
```

Ora salvate il programma, con

SAVE "ZANNA3" e VERIFY

e date il RUN. Dopo il solito inizio, scrivete:

Cosa devo fare ? grunt

- Non conosco il verbo 'grunt'.

Questo vuol dire: la parola GRUNT non è nel mio dizionario. Se la parola sconosciuta è al secondo posto (gli articoli non contano), la risposta è invece:

Cosa devo fare ? guarda il grunt

- Non conosco la parola 'grunt'.

Il che, implicitamente, significa che conosce la parola GUARDA (infatti, c'è nel dizionario). Ora, scriviamo una frase sensata:

Cosa devo fare ? apri la porta

- Non capisco.

Perchè non capisce? Perchè conosce le singole parole APRI e PORTA, ma non l'azione descritta dalle due parole combinate. Dovremo insegnargliela, ma più tardi. Per adesso, notate che ci sono alcune azioni già predisposte e funzionanti (le ho introdotte io, per facilitarvi il lavoro), ad esempio:

Cosa devo fare ? prendi la chiave

- Qui non ne vedo.

Ha ragione: non c'è nessuna chiave. Abbiamo infatti inserito la parola CHIAVE, ma non l'oggetto corrispondente. Dobbiamo ancora farlo.
Prima di passare ad occuparci degli oggetti, fate un esperimento: inserite un vocabolo fuori dell'ordine alfabetico e fate partire il programma. Guardate cosa succede.

La tavola degli oggetti

Perchè CHIAVE non sia una semplice parola, ma un oggetto che fa parte dell'avventura, dobbiamo inserire le caratteristiche di questo oggetto nell'apposita tavola, costituita da un'elenco di linee di DATA per ora, l'elenco è vuoto:

```
54970 rem
54980 rem - oggetti:
54990 rem
55000 data "fo": rem --- descrizioni e dati oggetti, dal n.1 ---
```

E' molto simile alla mappa: una serie di linee di DATA, terminate da "fo". Ogni linea ha questo aspetto:

```
55000 data "una chiave di sicurezza",40,4
```

Ci sono una descrizione e due numeri. La prima viene usata dal programma per descrivere l'oggetto (Vedo...). Il primo numero è il codice (nel dizionario) della parola che descrive l'oggetto (40 è il codice di CHIAVE), il secondo è il luogo in cui l'oggetto inizia il gioco.

Attenzione: la chiave è l'oggetto **numero 1**, in quanto è il primo dell'elenco. Non fate confusione con il **40**, che è il **codice** della parola CHIAVE. In altri termini: l'oggetto numero 1 risponde alla parola CHIAVE. Invece di scrivere CHIAVE nell'elenco oggetti, ne scrivo il codice, cioè 40. Dato che la chiave si trova in cantina, il secondo numero è 4 (ricordo che la cantina è il luogo numero 4).

Non c'è alcun ordine fisso per gli oggetti (alfabetico o altro), ma bisogna ricordarsi che sono identificati dalla loro posizione nell'elenco: il primo è il numero 1, il secondo il numero 2, ecc. Eventuali nuovi oggetti vanno dunque aggiunti in fondo, per non creare confusione. Tanto perchè lo sappiate, il programma elenca gli oggetti visibili nell'ordine in cui sono nell'elenco.
Dunque, ecco l'elenco dei nostri oggetti:

```
55000 data "una chiave di sicurezza",40,4
55010 data "dei microfilm arrotolati",41,3
55020 data "una porta blindata",60,2
55030 data "fo"
```

Ancora una volta, non dimenticate la chiusura "fo". Dato l'ordine in cui sono messi, gli oggetti hanno questi numeri, che serviranno per identificarli in tutto il resto del programma:

1) chiave di sicurezza.
2) microfilm arrotolati.
3) porta blindata.

Introdotto l'elenco degli oggetti, è il caso di salvare il programma:

SAVE "ZANNA4" e VERIFY

Spero che stiate usando, come vi ho consigliato, una cassetta corta con nastro da computer: si riavvolge velocemente, e dà più sicurezza. A proposito di sicurezza, è il caso di fare un SAVE "ZANNA4" anche sulla seconda cassetta vergine (o disco inizializzato) che avevate preparato (vero?) Salvando sulla seconda cassetta ogni tre versioni, e conservando la seconda cassetta in un posto diverso dalla prima, avete la massima sicurezza di non perdere tutto il vostro lavoro per uno stupido inconveniente (come il registratore che ha fame e decide di mangiarsi il nastro).

D'ora in poi, non vi dirò più di salvare il programma periodicamente. Se avete capito l'antifona, bene, se no affari vostri. Tanto, non sarò io a doverlo riscrivere daccapo.

---

date il RUN:

Sei in un atrio arredato con stile.
Vedo una porta blindata.

La porta c'è. Andiamo a vedere il resto:

Cosa devo fare ? b

Sei in una cantina ammuffita.
Vedo una chiave di sicurezza.

Come dicevo, le azioni PRENDI e LASCIA sono già predisposte, dunque è come fossero automatiche:

Cosa devo fare ? prendi la chiave

Fatto.

E possiamo verificare:

Cosa devo fare ? cosa

Possiedi:

- una chiave di sicurezza.

Tutto bene, dunque? Quasi:

Cosa devo fare ? a

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? prendi la porta

Fatto.

Sei in un atrio arredato con stile.

Cosa devo fare ? cosa

Possiedi:

- una chiave di sicurezza.
- una porta blindata.

Va be' che un agente segreto è sempre in buona forma fisica, ma portarsi sottobraccio una porta blindata mi pare eccessivo. Come si fa ad impedire che la porta blindata possa essere presa tranquillamente, come qualsiasi altro oggetto? Torniamo all'elenco degli oggetti:

```
55000 data "una chiave di sicurezza",40,4
55010 data "dei microfilm arrotolati",41,3
55020 data "una porta blindata",60,2
55030 data "fo"
```

L'ultimo numero di ogni linea, come già visto, indica il luogo iniziale dell'oggetto. Ecco tutti i luoghi possibili:

```
 1..98       il luogo corrispondente,  prendibile.
-1..-98      il luogo corrispondente,  non prendibile.
-99          il Limbo.
 0           trasportato  dall'avventuriero.
```

Quindi, se un oggetto si trova nel luogo zero, significa che è in possesso dell'avventuriero. Se si trova nel luogo -99, è nel Limbo, cioè in una specie di deposito fuori del gioco, da cui può essere ripescato quando si vuole.
Se infine il luogo di un oggetto è negativo, vuol dire che l'oggetto non può essere preso. Questo è il caso della nostra porta blindata, dunque correggiamo:

```
55000 data "una chiave di sicurezza",40,4
55010 data "dei microfilm arrotolati",41,3
55020 data "una porta blindata",60, -2
55030 data "fo"
```

e proviamo (RUN):

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? prendi la porta

- Non e' possibile.

Così va meglio. Però:

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? e

Sei nello studio dell'Ambasciatore.
Vedo dei microfilm arrotolati.

La porta blindata non blocca il passaggio! Per forza, come fa il programma a saperlo se non glielo spieghiamo? E' ora di passare all'azione, ed a questo è dedicato il prossimo capitolo.

Capitolo 8: una porta chiusa a chiave

Il programma deve conoscere le azioni che il giocatore può compiere, ciascuna delle quali è definita da una certa frase scritta in un certo luogo. L'effetto di un'azione consiste, in genere, nella stampa di un messaggio e nella modifica della posizione di uno o più oggetti.

Torniamo alla nostra porta blindata. Deve aprirsi con la frase:

APRI LA PORTA

ma solo se l'avventuriero possiede la chiave (che abbiamo messo in cantina). La prima cosa da fare è insegnare al programma a riconoscere la frase APRI LA PORTA. Dato che gli articoli sono ignorati, la frase si riduce in realtà a:

APRI PORTA

C'è un altro punto da considerare: la frase deve valere soltanto nell'atrio, cioè nel luogo 2, dove si trova la porta. Riassumendo:

Luogo:	Prima parola:	Seconda parola:
ATRIO	APRI	PORTA

Sostituiamo le parole con i rispettivi codici: l'atrio è il luogo numero 2, APRI e PORTA hanno rispettivamente codice 20 e 60 nel dizionario:

Luogo:	Prima parola:	Seconda parola:
2	20	60

Portiamo a due cifre tutti i numeri. In questo caso, è solo il numero del luogo che richiede una modifica:

Luogo:	Prima parola:	Seconda parola:
02	20	60

e finalmente uniamo i tre numeri, ottenendo un unico numero di sei cifre:

022060

Questo è il **codice della frase** APRI LA PORTA nel luogo 2 (l'atrio). Dato che si tratta di un numero (e non di una stringa), lo zero iniziale può essere tralasciato (insomma, era inutile aggiungerlo):

22060

Conviene, comunque, abituarsi a fare questo lavoro con sei cifre: due per il luogo, due per la prima parola e due per la seconda parola, togliendo alla fine lo zero o gli zeri iniziali di troppo. E' meno facile sbagliarsi.

Che ce ne facciamo di questo numero? Scommetto che avete già indovinato: lo introduciamo nell'apposita **tavola delle azioni**. Al momento, ci sono alcune azioni già predisposte:

```
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 data "fa": rem  --- ricordarsi  l'ordine numerico!!!  ---
```

Ogni azione è indicata con due numeri: il **codice della frase** ed il **numero dell'azione**. Il primo non è altro che il numero che abbiamo appena preparato, il secondo merita un'ulteriore spiegazione. Per il momento, diciamo che è un numero compreso tra 8 e 160 e che conviene usare il primo numero libero. Dato che non ne abbiamo ancora impiegato nessuno, usiamo 8.
L'azione va introdotta nella tavola rispettando l'**ordine numerico** dei codici di frase (il numero di azione non conta). Se ci si sbaglia, niente paura: il programma non parte e segnala un errore, indicando i primi due codici fuori ordine (esattamente come succede con il dizionario).


DAF

Dunque, introduciamo la nostra azione numero 8 (APRI LA PORTA, nell'atrio):

```
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 data 22060,8
50030 data "fa": rem  --- ricordarsi  l'ordine  numerico!!!  ---
```

e facciamo partire il programma (RUN):

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? apri la porta

8
```

Che vuol dire? Semplice: il programma ha riconosciuto la frase ed ha tentato di eseguire l'azione numero 8 (che dobbiamo ancora scrivere). Verifichiamo che l'azione funzioni solo nell'atrio:

```
Cosa devo fare ? n

Sei in un'ampia camera da letto.

Cosa devo fare ? apri la porta

- Non capisco.
```

Infatti, in camera da letto non c'è nessuna porta. Ora che la frase viene riconosciuta, e solo nel luogo voluto, è il momento di occuparsi dell'azione vera e propria.

Passiamo all'azione

Il numero 8 che viene stampato in risposta alla frase APRI LA PORTA (pronunciata nell'atrio) viene da qui:

```
6390 rem 8:
6400 print 8:return
6590 rem 9:
6600 print 9:return
6790 rem 10:
6800 print 10:return
....
```

In risposta alla nostra frase, il programma ha eseguito automaticamente un GOSUB 6400. Se avessimo messo 9 come numero dell'azione, avrebbe eseguito un GOSUB 6600, con 10 un GOSUB 6800, e così via fino all'azione numero 160, che fa eseguire un GOSUB 36800 (vedi il listato del Modulo Base nell'Appendice A).
Insomma, l'azione numero 8 consiste nell'eseguire le linee BASIC che vanno dalla 6400 fino alla prima istruzione RETURN, che termina l'azione.
Non è troppo chiaro? Modificate la linea 6400:

```
6400 print "Ehi, funziona!": return
```

RUN

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? apri la porta

Ehi, funziona!

Visto? Possiamo costruire l'azione numero 8 come ci pare e piace. Possiamo ottenere tutto quello che vogliamo. Il problema, naturalmente, è sapere cosa vogliamo.
La prima cosa da fare è verificare se l'avventuriero ha la chiave. In caso contrario, la porta non può essere aperta. Diciamolo in modo più formale:

- Se non ha la chiave, stampa "Non hai la chiave" e termina l'azione.
- Altrimenti, apre la porta e termina l'azione.

Come si fa a sapere se la chiave è in possesso dell'avventuriero? C'è un array che contiene il luogo di ciascun oggetto. Se non sapete cos'è un array (è roba che si mangia?), niente paura: per usarlo non occorre sapere com'è fatto. Basta sapere che:

L(1) è il luogo dell'oggetto 1
L(2) è il luogo dell'oggetto 2
L(3) è il luogo dell'oggetto 3
(ecc.)

Per capire meglio la cosa, fate partire il programma (RUN), poi fermatelo con il solito RUN/STOP-RESTORE (e poi SHIFT-C=) e scrivete:

print l(1) (l è una elle minuscola)
4

Dice che il luogo dell'oggetto 1 è 4, cioè che l'oggetto 1 si trova nel luogo 4. E' vero? Sì, l'oggetto 1 è la chiave (vedi cap.7) e il luogo 4 è la cantina. Adesso lo freghiamo noi: gli cambiamo il luogo dell'oggetto 1:

l(1)=0

e lo facciamo ripartire, facendo:

goto 760

Sei in atrio arredato con stile.
Uedo una porta blindata.

Cosa devo fare ? cosa

Possiedi:

- una chiave di sicurezza.

Cosa abbiamo fatto? Abbiamo messo l'oggetto 1 (la chiave) nel luogo zero. Ma il luogo zero vuol dire "trasportato dall'avventuriero", quindi abbiamo dato la chiave in mano all'agente segreto! Con questo sistema, possiamo manipolare tutti gli oggetti del gioco a nostro piacimento (incredibile! con queste diavolerie moderne, chissà dove andremo a finire).

Riassumendo, per sapere se l'avventuriero ha la chiave (in altri termini: se la chiave è trasportata), basta l'istruzione BASIC:

```
if l(1)=0 then...
```

Cioè: se (IF) l'oggetto 1 (la chiave) si trova nel luogo zero (trasportata), allora (THEN) esegue il resto della linea, altrimenti lo salta. Tornando alla nostra azione 8 in costruzione:

```
6390 rem 8:
6400 if l(1)<>0 then print "Non hai la chiave.":return
6410 print "Ok":return
```

Dunque: se non ha la chiave (il luogo dell'oggetto 1 è diverso da zero, dunque non è trasportato), stampa "Non hai la chiave" e termina l'azione (RETURN). In caso contrario (quindi, se ha la chiave), prosegue alla linea successiva, che stampa "Ok" e termina l'azione. Proviamolo (RUN):

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? apri la porta

Non hai la chiave.

Cosa devo fare ? b

Sei in una cantina ammuffita.
Vedo una chiave di sicurezza.

Cosa devo fare ? prendi la chiave

Fatto.
```

```
Cosa devo fare ? a

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? apri la porta

Ok
```

Funziona. Per adesso si limita a stampare "Ok", ma funziona: apre la porta solo se ha la chiave.

Adesso bisogna far aprire effettivamente la porta. Sarebbe una buona cosa trasformare "una porta blindata" in "una porta blindata aperta", ma il Modulo Base non consente di cambiare la descrizione di un oggetto. E allora?

Allora, c'è un trucco molto semplice: aggiungiamo un nuovo oggetto all'elenco:

```
55000 data "una chiave di sicurezza",40,4
55010 data "dei microfilm arrotolati",41,3
55020 data "una porta blindata",60,-2
55030 data "una porta blindata aperta",60,-99
55040 data "fo"
```

Ricordo che i nuovi oggetti vanno aggiunti in fondo all'elenco, per non alterare il numero dei precedenti. Abbiamo aggiunto l'oggetto numero 4, con descrizione "una porta blindata aperta", che risponde alla parola PORTA (codice 60 nel dizionario) e che si trova nel luogo -99.

Se l'avete dimenticato, il luogo -99 è il Limbo, cioè il "deposito" degli oggetti fuori gioco. All'inizio dell'avventura, l'oggetto 4 non c'è.

Dato che siete svegli, avrete già capito il trucco: per aprire la porta, scambiamo due oggetti: facciamo sparire l'oggetto 3 (la porta chiusa) e facciamo apparire l'oggetto 4 (la porta aperta). Si fa in un attimo:

```
6390 rem 8:
6400 if l(1)<>0 then print "Non hai la chiave.":return
6410 print "Ok":l(3)=-99:l(4)=-2: return
```

(Notate il segno meno del 2, perchè la porta non è prendibile). Date il solito RUN, andate a prendervi la chiave (B, PRENDI LA CHIAVE, A) e provate:

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? apri la porta

Ok

Sei in un atrio arredato con stile.
Vedo una porta blindata aperta.

Noi sappiamo che sono stati scambiati due oggetti (il 3 ed il 4), ma il giocatore non se ne accorge: per lui è sempre il medesimo oggetto (la porta), che ha cambiato stato (da chiusa ad aperta).

E la porta funziona. Se vi sembra che ci sia voluto tanto lavoro, è solo perchè ho descritto tutto nei minimi particolari: in realtà, sono soltanto quattro righe di programma (6400, 6410, 50020 e 55030).

Resta una cosa da fare: la porta non blocca il passaggio! Bisogna fare in modo che lo spostamento dall'atrio allo studio sia permesso solamente se la porta è aperta.
Quando interviene la porta? Quando il giocatore fa EST (o E) dall'atrio. Dobbiamo quindi scrivere un'azione che risponde alla frase EST pronunciata nell'atrio. Ma questa non va ad interferire con il regolare funzionamento della direzione EST? No, perchè la frase EST pronunciata in un certo luogo è diversa dalla frase EST generica (valida in tutti i luoghi), ed ha la precedenza su questa. Possiamo dunque costruire il codice di azione:

Luogo:	Prima parola:	Seconda parola:
ATRIO	EST	(nessuna)
02	03	00

Due note: il codice di EST è 3, come si vede dal vocabolario (linea 40040), e lo portiamo a due cifre (03). Se la frase dev'essere composta di una sola

parola, si indica 00 come seconda parola. Il codice dell'azione EST nell'atrio è dunque:

020300

cioè 20300 (gli zeri a sinistra si possono togliere). Mettiamolo nella tavola delle azioni, indicando il numero dell'azione da eseguire in risposta alla frase EST nell'atrio (prendiamo la prima libera: la numero 9):

```
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 data 20300,9 ,22060,8
50030 data "fa": rem --- ricordarsi l'ordine numerico!!! ---
```

Era la stessa cosa scrivere una nuova linea di DATA, invece di aggiungerla in una linea preesistente. L'importante è rispettare l'ordine numerico. Adesso sappiamo (se avete dubbi, provatelo) che la frase EST, scritta nell'atrio, causa l'esecuzione della routine di azione numero 9 (linea 6600).

Cosa deve fare l'azione numero 9? Semplice:

- Se la porta è chiusa, stampa "La porta è chiusa" e termina l'azione.
- Se la porta è aperta, lascia passare nello studio e termina l'azione.

Per sapere se la porta è chiusa, basta guardare il luogo dell'oggetto 3 (la porta chiusa): se l'oggetto 3 è nel luogo -2 (l'atrio, '-' perchè non prendibile), la porta è ancora chiusa, se è nel luogo -99 la porta è stata aperta. La stessa cosa, a rovescio, vale per l'oggetto 4 (la porta aperta). Infatti, come ricordate, l'apertura della porta consiste nel far sparire l'oggetto 3 (la porta chiusa) ed apparire l'oggetto 4 (porta aperta). Dunque:

```
6590 rem 9:
6600 IF l(3)=-2 then print "La porta e' chiusa.":return
6610 ???
```

(attenti al solito segno meno in -2). Già, cosa vuol dire "lascia passare

nello studio? Vuol dire spostare l'avventuriero nello studio, cioè nel luogo 3. Ma questo, se ricordate, lo sappiamo già fare: basta mettere il numero del nuovo luogo nella variabile LU (vedi Cap. 6). Quindi:

```
6590 rem 9:
6600 IF l(3)=-2 then print "La porta e' chiusa.":return
6610 lu=3:return
```

Proviamo: fate RUN ed andate a prendervi la chiave, poi tornate nell'atrio:

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? e

La porta e' chiusa.

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? apri la porta

Ok

Sei in un atrio arredato con stile.
Vedo una porta blindata aperta.

Cosa devo fare ? e

Sei nello studio dell'Ambasciatore.
Vedo dei microfilm arrotolati.
```

Finalmente, ci siamo. La sequenza principale è a posto e funziona. Adesso non resta che arricchirla con un po' di dettagli.
Nel prossimo capitolo, approfondirò anche le possibilità offerte dalle varie combinazioni possibili per il codice di azione. Una sintesi di tutto questo si trova, come al solito, nell'Appendice B. Una volta finito il libro, potrete usare l'Appendice B come guida di riferimento, senza dover sfogliare molte pagine alla ricerca delle informazioni che vi servono.


DAF
ABC

Capitolo 9: i dettagli contano

La qualità di un'avventura dipende molto dal livello di dettaglio. Un buon autore dedica parecchio tempo a rifinire ed aggiungere varianti, aiutandosi con i suggerimenti forniti dai collaudatori a cui fa provare il gioco.

Come dicevo, la sequenza principale è a posto e funziona. Ma l'avventura che ne risulta non è certo interessante: il giocatore deve sorbirsi un'infinità di "non capisco", e le uniche azioni valide sono quelle che portano direttamente alla soluzione dell'avventura. Purtroppo, ci sono in circolazione troppe avventure che non si discostano molto da questo sistema: o si fa l'azione giusta, o niente.
Contrariamente a quanto sembrano pensare alcuni autori, perchè un'avventura sia divertente non serve che abbia una mappa particolarmente estesa, o che presenti problemi difficilissimi. Deve invece essere molto curata nei dettagli, perchè sono proprio questi a dare al giocatore l'illusione di vivere realmente dentro l'avventura, in un universo fantastico che voi avete creato.
C'è però un prezzo da pagare: la rifinitura dei dettagli è la parte più laboriosa dell'intero lavoro, e potete aspettarvi di impiegare oltre la metà del tempo complessivo. Per essere sicuri di ottenere un buon risultato, ci sono due cose da fare:

- Aggiungere tutti i dettagli che vi vengono in mente.
- Far collaudare l'avventura, ed aggiungere nuovi dettagli.

Iniziamo dalla prima: cosa manca nella nostra micro-avventura? Per prima cosa, il finale. E' ovvio che ci vuole una qualche forma di conclusione.

Gran finale

Non cadete nell'altro errore tipico degli sceneggiatori trascurati: il finale insulso del tipo "congratulations, you win." (bravo, hai vinto). Dopo che uno ha speso sangue, sudore e lacrime per risolvere l'avventura, ha diritto ad

aspettarsi qualcosa di più gratificante. Direi che merita almeno una decina di righe (meglio se di più) dedicate all'apoteosi della sua impresa.
Dato però che non ho intenzione di scrivere una vera e propria avventura, mi limiterò a darvi uno spunto, che completerete voi se vorrete espandere l'esempio in un'avventura vera e propria (un ottimo e divertente esercizio).

Ma, prima di tutto, dove mettiamo il finale? Quando l'agente trova i microfilm? No, troppo presto: potrebbe ancora essere colto in flagrante. Bisogna uscire sani e salvi dall'appartamento, e solo allora la missione può dirsi compiuta.
Come facciamo ad impedire che l'avventuriero esca dall'appartamento prima di aver trovato i microfilm? Non c'è alcun motivo per impedirglielo, soltanto la sua missione fallisce miseramente se sceglie di farlo. E, si sa, un agente segreto che non sia in grado di compiere una missione è del tutto inutile, anzi è un intralcio di cui liberarsi al più presto...
Dunque, per concludere la missione occorre uscire dall'appartamento. L'uscita è ad ovest dell'atrio, e non occorre che la mettiamo nella mappa, perchè basta introdurre l'azione "Ovest" nel luogo 2 (l'atrio, appunto), così:

Luogo:	Prima parola:	Seconda parola:
ATRIO	OVEST	(nessuna)
02	04	00

Da cui:

020400, cioè 20400

che facciamo corrispondere alla prima azione libera, la numero 10:

```
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 data 20300,9, 20400,10 ,22060,8
50030 data "fa": rem  --- ricordarsi  l'ordine  numerico!!!  ---
```

Ora si tratta di scrivere l'azione numero 10. Se l'agente ha trovato i microfilm, la missione è riuscita, altrimenti è miseramente fallita. Per evitare di uscire per errore, diamo al giocatore la possibilità di confermare o annullare la sua scelta:

```
6790 rem 10:
6800 a$="Vuoi lasciare l'appartamento":gosub 1600
6810 if not (a) then return
6820 print:if l(2)=0 goto 6850:rem ok
6830 print "Fedeli agli ordini, i tuoi colleghi"
6840 print "eliminano un agente incapace.":goto 4000
6850 gosub 1500:rem <spazio>
6860 poke 53280,5:print:print "- Missione compiuta! -"
6870 print "Hai salvato il tuo Paese e ti"
6880 print "sei guadagnata la promozione a:":print
6890 print "          ******":print
6900 print "(il tuo nuovo titolo e' cosi' segreto"
6910 print "che non posso nemmeno nominarlo).":print
6920 print "Congratulazioni, ******!":print:end
```

C'è parecchio da dire. Iniziamo dalle linee 6800-6810. La 6800 chiama la subroutine 1600, che si occupa di porre una domanda, e torna solo quando il giocatore ha risposto "si" o "no" (o almeno l'iniziale "s"o "n"). La domanda da porre va messa nella variabile **a$** (senza spazi o punto di domanda). Al ritorno dalla subroutine, la variabile logica **a** conterrà il valore vero se la risposta era "si", falso se era "no". La linea 6810 significa dunque: se la risposta era "no" (NOT vero è evidentemente falso), ritorna senza fare nulla.

Se possiede i microfilm, cioè se il luogo dell'oggetto 2 (i microfilm) è zero (trasportati), la linea 6820 salta alla 6850, dove viene stampato il finale positivo (missione compiuta), preceduto da un "premi <spazio> per continuare" (che si ottiene con un GOSUB 1500) ed un cambio di colore di fondo. Alla fine, un END conclude il programma.

Se invece non ha i microfilm, il programma prosegue diritto alla 6830, dove stampa un messaggio di fallimento, e poi salta alla 4000. La routine che inizia alla linea 4000 è quella del finale negativo (corrispondente di solito, anche se non sempre, alla morte dell'avventuriero). Al momento, la routine fa ben poco:

```
3970 rem
3980 rem - morto -
3990 rem
4000 gosub 1500:poke 53280,2:rem <sp>
4010 rem --- mettere qui il finale negativo ---
4450 a$="Vuoi giocare ancora":gosub1600:if(a)then  run:rem riparte
4460 print:print:print"Ciao!":print:end
```

Si tratta di introdurre un messaggio, alla fine del quale il programma chiede se ripartire dall'inizio o smettere di giocare (linea 4450, notate l'uso della subroutine 1600).
Anche qui, ci sta bene un'adeguata conclusione, di cui mi limito a fornire un breve spunto:

```
3970 rem
3980 rem - morto -
3990 rem
4000 gosub 1500:poke 53280,2:rem <sp>
4010 print:print "La tua missione e' fallita."
4020 print "Di te non restera' che un"
4030 print "polveroso fascicolo, presto"
4040 print "dimenticato.":print
4450 a$="Vuoi giocare ancora":gosub1600:if(a)then  run:rem riparte
4460 print:print:print"Ciao!":print:end
```

Ora, collaudiamo il tutto (RUN):

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? o

Vuoi lasciare l'appartamento ? no

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? o
```


TOP SECRET
DAF

Vuoi lasciare l'appartamento ? si

Fedeli agli ordini, i tuoi colleghi
eliminano un agente incapace.

- Premi <spazio> per continuare -

La tua missione e' fallita.
Di te non restera' che un
polveroso fascicolo, presto
dimenticato.

Vuoi giocare ancora ?

Ora, riprovate il tutto dopo aver preso i microfilm. Se non avete voglia di fare il giro per prendere i microfilm, usate il solito trucco:

- fermate il programma con RUN/STOP-RESTORE (e poi SHIFT-C=).
- prendete i microfilm con I(2)=0.
- fate ripartire il programma con goto 760.

Con i microfilm in mano, il finale e' diverso:

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? o

Vuoi lasciare l'appartamento ? si

- Premi <spazio> per continuare -

- Missione compiuta! -
Hai salvato il tuo Paese e ti
sei guadagnata la promozione a:

****

**(il tuo nuovo titolo e' cosi' segreto
che non posso nemmeno nominarlo).**

Congratulazioni, ***!**

Ed il finale è a posto. Noterete, scrivendo avventure, che una delle cose più noiose è la necessità di controllare l'effetto dei PRINT sul video: bisogna, ogni volta, far ripartire il programma, procurarsi gli oggetti, andare nel luogo giusto, ecc. C'è un sistema più semplice: chiamare direttamente le linee che eseguono la stampa. Per esempio, volendo controllare il finale negativo basta fare un GOSUB (a programma fermo) alla prima linea di PRINT:

```
gosub 4010
```

**La tua missione e' fallita.
Di te non restera' che un
polveroso fascicolo, presto
dimenticato.**

Questa tecnica è comodissima, anche se a volte può dare messaggi di errore del tipo ?BAD SUBSCRIPT ERROR se non avete prima fatto girare il programma. In ogni caso, non c'è nessuna conseguenza negativa.

Guardarsi bene in giro

Una delle azioni fondamentali in ogni avventura è GUARDA. E' lecito aspettarsi che un giocatore guardi ogni oggetto, alla ricerca di informazioni o indizi. Sarebbe ingiusto e fastidioso rispondere sempre "non noto nulla di particolare". E' invece il caso di prevedere un'azione GUARDA per tutti, o quasi tutti, gli oggetti. Per esempio, possiamo usare la prima azione libera (la numero 11) per GUARDA LA CHIAVE:

Luogo:	Prima parola:	Seconda parola:
(dovunque)	GUARDA	CHIAVE

E' ovvio che non possiamo limitare la frase GUARDA LA CHIAVE ad un solo

luogo, dato che la chiave può essere presa e portata in giro per l'appartamento. Per indicare che una frase è valida in tutti i luoghi, si indica zero come numero di luogo:

Luogo:	Prima parola:	Seconda parola:
00	10	40

Cioè 1040. Dunque, aggiorniamo la tavola delle azioni:

```
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4, 1040,11 ,1099,-4,1100,5,1200,6,1300,7
50020 data 20300,9,20400,10,22060,8
50030 data "fa": rem  --- ricordarsi  l'ordine  numerico!!!  ---
```

E mettiamo un messaggio come azione 11:

```
6990 rem 11:
7000 print "E' una Yale di sicurezza.":return
```

Ora, proviamo (RUN), andando per prima cosa in cantina a prendere la chiave e poi scrivendo:

Cosa devo fare ? guarda la chiave

E' una Yale di sicurezza.

Perfetto. O no? Riproviamo senza la chiave:

Cosa devo fare ? guarda la chiave

E' una Yale di sicurezza.

Ahi! Non si accorge che non abbiamo la chiave. Bisogna modificare l'azione 11 per verificare che la chiave sia trasportata. Dato però che questo è un caso molto comune, ho previsto un sistema automatico per fare il

controllo: se il numero di azione è negativo, il Modulo Base controlla che la seconda parola della frase rappresenti un oggetto presente nel luogo o trasportato dall'avventuriero. Insomma, basta un segno meno:

```
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4, 1040,-11 ,1099,-4,1100,5,1200,6,1300,7
50020 data 20300,9,20400,10,22060,8
50030 data "fa": rem --- ricordarsi l'ordine numerico!!! ---
```

Riprovando (RUN) senza la chiave:

Cosa devo fare ? guarda la chiave

- Qui non ne vedo.

mentre, con la chiave presa o visibile, funziona tutto come prima. Perchè questo controllo non viene fatto automaticamente su tutte le frasi? Per consentire la massima libertà in frasi come "CERCA LA CHIAVE" , a cui si può rispondere "cercatela da te" anche se non è presente, tanto per fare un esempio.

Lascio a voi il compito di aggiungere l'azione GUARDA per tutti gli altri oggetti, ricordandovi di mettere un meno davanti al numero di azione quando si tratta di un oggetto trasportabile.

Altre azioni da aggiungere? Quando avete finito le idee, prendete qualche vostro amico, nominatelo ufficialmente "collaudatore di primo livello" e fatelo giocare, prendendo note senza dire nulla: in mezz'ora avrete un notes pieno di frasi a cui non avevate minimamente pensato. Armatevi di pazienza, ed introducetele (ampliando il vocabolario con le nuove parole). E' così che si costruisce una bella avventura.

Azioni per tutti i gusti

Avrete notato che ci sono varie possibili combinazioni per il codice di

azione. Ve le descrivo una per una, ricordandovi che potrete poi consultare l'Appendice B per ricordarvele quando vi servono.

Con frasi di una sola parola, sapete già che basta indicare 00 come codice della seconda parola. ad esempio, per NORD nel luogo 3:

Luogo:	Prima parola:	Seconda parola:
03	01	00

quindi il codice è 30100. Ci sono due possibili combinazioni, che elenco in ordine di precedenza:

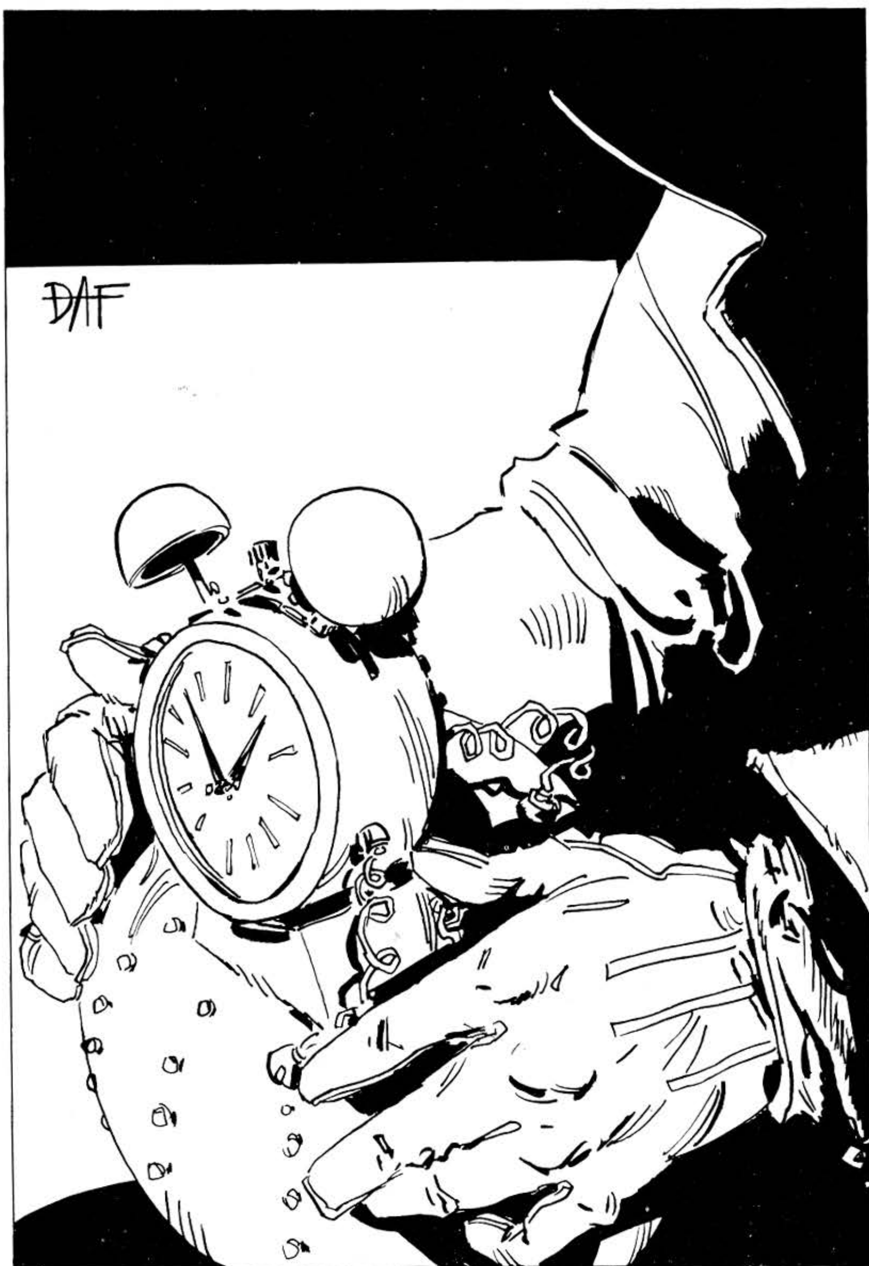
1 - Parola valida in un certo luogo.
2 - Parola valida ovunque.

La precedenza significa che il programma guarda per prima cosa se è stata prevista l'azione come valida nel luogo corrente dell'avventuriero (come il nostro EST nell'atrio). Se non la trova, guarda se l'azione è stata prevista come valida in tutti i luoghi (come un EST generico). Se non la trova ancora, stampa "non capisco".
Con le frasi di due parole, ci sono quattro combinazioni, che sono esaminate in quest'ordine:

1 - Prima parola + seconda parola, valide in un certo luogo.
2 - Prima parola + parola qualunque, valide in un certo luogo.
3 - Prima parola + seconda parola, valide ovunque.
4 - Prima parola + parola qualunque, valide ovunque.

Il caso 1 è ben noto (APRI LA PORTA, nell'atrio), come pure il caso 3 (GUARDA LA CHIAVE, dovunque). Per rendere l'azione valida ovunque, basta mettere zero come numero di luogo.
I casi 2 e 4 (che differiscono tra loro solo per il numero di luogo, che è zero nel caso 4) sono un po' particolari: l'azione viene eseguita comunque, purchè ci sia una seconda parola valida, cioè esistente nel dizionario. Un esempio tipico (che avete usato finora senza saperlo) è PRENDI, che viene eseguita qualunque sia la seconda parola che segue. E' utile sapere che in questo caso la variabile OG contiene il numero dell'oggetto nominato, o zero se la seconda parola non indica un oggetto.

E' istruttivo esaminare come funzionano le azioni di PRENDI (linea 5200) e LASCIA (linea 5400), notando che LASCIA può essere eseguita anche su oggetti non presenti, mentre PRENDI ha numero di azione negativo e viene quindi chiamata solo se la seconda parola indica un oggetto presente o trasportato.
Sempre nell'appendice B, trovate le principali variabili usate dal programma, che possono esservi utili nelle vostre routine di azione. Ad esempio, p1$ e p2$ contengono rispettivamente la prima e la seconda parola, mentre c1 e c2 sono i rispettivi codici.
Se avete le idee un po' confuse, non perdetevi d'animo. Cominciate a scrivere una piccola avventura, e vedrete che molte cose si chiariranno man mano. E mi raccomando: fate più errori possibile. Sbagliando s'impara!



Capitolo 10: una bomba ad orologeria

Il Modulo Base permette anche di inserire nell'avventura il fattore tempo, facendo accadere un dato evento in un momento prefissato o in rapporto ad una o più altre azioni compiute dall'avventuriero.

Un'avventura può diventare più interessante se si tiene conto del passaggio del tempo. Non il tempo che il giocatore passa a pensare quale azione compiere, perchè non è il caso di aggiungere l'angoscia di una risposta veloce ai molti problemi che già il poveretto si trova a dover affrontare. Parlo invece del tempo inteso come numero di mosse compiute: un esempio ormai abusato è quello della lampada la cui batteria si scarica dopo un certo numero di mosse, lasciando l'avventuriero al buio.
Nella nostra avventura, mi pare logico porre un tempo limite, come avverrebbe in condizioni reali. Non è il caso di renderlo molto stringente: diciamo 25 mosse, al termine delle quali l'ambasciatore torna a casa.

Il Modulo Base è già predisposto per tener conto del passaggio del tempo. Ci sono cinque timer usabili a piacimento (t1, t2, t3, t4, t5), che funzionano in questo modo:

- Un timer che sta a zero è inattivo, cioè non ha effetto.
- Un timer contenente un numero positivo è in funzione.

Se un timer è in funzione, scende di uno ad ogni mossa, e chiama una certa routine (sempre ad ogni mossa), finchè non arriva a zero. Per la precisione:

- Il Timer 1 esegue un GOSUB 3000 ad ogni mossa in cui è attivo.
- Il Timer 2 esegue un GOSUB 3200 ad ogni mossa in cui è attivo.
- Il Timer 3 esegue un GOSUB 3400 ad ogni mossa in cui è attivo.
- Il Timer 4 esegue un GOSUB 3600 ad ogni mossa in cui è attivo.
- Il Timer 5 esegue un GOSUB 3800 ad ogni mossa in cui è attivo.

Dato che i timer contano alla rovescia, per farli partire occorre metterci un numero positivo. Se vogliamo limitare il gioco a 25 mosse, possiamo

fare così:

```
4950 lu=2:t1=25:print:return
```

La linea 4950, come potete controllare sul vostro listato, è l'ultima dell'introduzione. Abbiamo dunque fatto partire il Timer 1 (t1), che funzionerà per 25 mosse prima di arrivare a zero e fermarsi. Provate a fare RUN:

Sei in un atrio arredato con stile.
Vedo una porta blindata.

T1

E' apparso un "T1". Se provate a far passare un po' di tempo, ad esempio spostandovi tra una stanza e l'altra, vedrete che dopo 25 mosse il "T1" scomparirà: a forza di togliere 1, il timer è arrivato a zero e si è fermato. La stampa del "T1" è stata provocata da questa routine:

```
2990 rem t1:
3000 print:print "T1":return
```

Come al solito, possiamo modificare a piacimento la routine. Nel nostro caso, dobbiamo semplicemente ignorare il timer (non far niente) se non è arrivato a zero, mentre in quest'ultimo caso accade l'inevitabile:

```
2990 rem t1:
3000 if t1>0 then return
3010 print:print "L'ambasciatore ritorna a casa."
3020 print "E' la fine!":goto 4000
```

Dando il RUN, noterete che non appare più il "T1" (per forza, abbiamo tolto il PRINT). Dobbiamo ora verificare che dopo 25 mosse torni l'ambasciatore. Per non aspettare 25 mosse, conviene usare il solito trucco:

- Fermare il programma con RUN/STOP-RESTORE (e poi SHIFT-C=).
- Abbreviare il tempo rimasto con t1=4.
- Ripartire con goto 760.

Ora, il timer arriverà a zero in sole 3 mosse (la quarta è il goto 760), e stamperà:

L'ambasciatore ritorna a casa.
E' la fine!

- Premi <spazio> per continuare -

seguito dal solito finale negativo (ottenuto con il goto 4000). Nessuno proibisce di arricchire la vicenda con eventi collegati a certi valori del timer, ad esempio con una sequenza del tipo:

```
3000 if t1=5 then print "Sento un'auto che arriva.":return
3010 if t1=3 then print "Sento sbattere una portiera.":return
3020 if t1=2 then print "Sento dei passi.":return
```

seguita dalle istruzioni viste in precedenza. La presenza di timer attivi (cioè diversi da zero) rallenta un poco il programma, ma vivacizza il gioco.

Un esempio più complesso

Vediamo un altro esempio di uso dei timer. Complichiamo la vicenda, mettendo la chiave in una cassaforte che si può aprire soltanto con una bomba ad orologeria mascherata da sveglia. Abbiamo bisogno di questi oggetti:

- Una sveglia di buona marca.
- Una solida cassaforte.
- Una cassaforte sventrata.
- Un cumulo di macerie.

e delle parole (a cui assegno già i codici):

- CARICA	21
- SVEGLIA	42
- CASSAFORTE	43
- CUMULO, MACERIE	61

Modifichiamo dunque il dizionario:

```
39970 rem
39980 rem - dizionario:
39990 rem
40000 data "?",13,"a",5,"agli",7,"al",7,"all",7
40010 data "alla",7,"alle",7,"allo",7,"alto",5
40020 data "b",6,"basso",6
40025 data "carica",21,"cassaforte",43,  "chiave",40
40030 data "col",7,"con",7,"cosa",13,  "cumulo",61
40040 data "e",3,"est",3
40050 data "gli",7,"guarda",10
40060 data "i",7,"il",7,"inventario",13
40070 data "l",7,"la",7,"lascia",9
40080 data "le",7,"lo",7,"load",12
40085 data "macerie",61 ,"microfilm",41
40090 data "n",1,"nord",1
40100 data "o",4,"ovest",4
40105 data "porta",60
40110 data "posa",9,"prendi",8
40120 data "riprendi",12
40130 data "s",2,"sali",5,"salva",11
40140 data "save",11,"scendi",6,"sud",2,  "sveglia",42
40150 data "un",7,"una",7,"uno",7
40160 data "w",4
40170 data "fd": rem --- ricordarsi  l'ordine  alfabetico!!!  ---
```

e la tavola degli oggetti:

```
55000 data "una chiave di sicurezza",40 ,-99
55010 data "dei microfilm arrotolati",41,3
55020 data "una porta blindata",60,-2
55030 data "una porta blindata aperta",60,-99
55040 data "una sveglia di buona marca",42,1
55050 data "una solida cassaforte",43,-4
55060 data "una cassaforte sventrata",43,-99
55070 data "un cumulo di macerie",61,-99
55080 data "fo"
```

La sveglia si trova, all'inizio, in camera da letto (luogo 1), e la cassaforte sta in cantina (luogo 4, ma col segno meno, perchè non è prendibile). La cassaforte sventrata ed il cumulo di macerie sono, ovviamente, nel Limbo (-99). Notate la modifica alla linea 55000: la chiave non è più in cantina, ma nel Limbo; apparirà solo una volta distrutta la cassaforte.
Provate (RUN), e verificate che la sveglia sia in camera da letto e che si possa prendere, e che la cassaforte sia in cantina, e non prendibile.
Si tratta ora di usare un timer per la sveglia/bomba. Dato che il timer 1 è già impegnato, useremo il timer 2. L'azione CARICA LA SVEGLIA farà partire il timer con un tempo abbastanza breve, diciamo 5 mosse. L'azione GUARDA LA SVEGLIA ci dirà se è carica o meno. Entrambe le azioni sono valide dovunque, richiedono che l'oggetto (la sveglia) sia presente (numero di azione negativo), ed hanno rispettivamente codice 00-21-42 (dovunque, CARICA, SVEGLIA), cioè 2142, e 00-10-42 (dovunque, GUARDA, SVEGLIA), cioè 1042. Le prime azioni libere sono la 12 e la 13, dunque:

```
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4,1040,-11, 1042,-13 ,1099,-4,1100,5,1200,6,1300,7
50015 data 2142,-12
50020 data 20300,9,20400,10,22060,8
50030 data "fa": rem  --- ricordarsi  l'ordine  numerico!!!  ---
```

Scriviamo l'azione 12 (CARICA LA SVEGLIA):

```
7190 rem 12:
7200 if t2>0 then print "Gia' fatto.":return
7210 print "Cric, cric, cric...":t2=5:return
```

e la 13 (GUARDA LA SVEGLIA):

```
7390 rem 13:
7400 if t2=0 then print "E' scarica.":return
7410 print "Ticchetta.":return
```

Nell'azione 12, è necessario controllare che la sveglia non sia già carica, altrimenti il timer viene rimesso a 5 ogni volta (e non è realistico, anche

se non pregiudica il gioco).
Ora fate RUN, andate in camera da letto, e provate:

```
Sei in un'ampia camera da letto.
Vedo una sveglia di buona marca.

Cosa devo fare ? guarda la sveglia

E' scarica.

Sei in un'ampia camera da letto.
Vedo una sveglia di buona marca.

Cosa devo fare ? prendi la sveglia

Fatto.

Sei in un'ampia camera da letto.

Cosa devo fare ? carica la sveglia

Cric, cric, cric...

Sei in un'ampia camera da letto.

T2

Cosa devo fare ? carica la sveglia

Gia' fatto.

T2

Cosa devo fare ? guarda la sveglia

Ticchetta.

T2
```


DAF

Ed il "T2" continua ad apparire per altri due turni, dopo di che scompare: il timer 2 è arrivato a zero. Adesso, occupiamoci di questo timer:

- Ad ogni turno, scrive un "tic, tac" se ha la sveglia in mano (luogo zero).
- A zero, la bomba esplode, e ci sono tre casi:
    - E' nello stesso luogo dell'avventuriero: ovvia conseguenza.
    - E' in cantina: distrugge la cassaforte ed appaiono macerie e chiave.
    - E' in un altro luogo: appaiono solo le macerie.

Notate che per sapere se la sveglia (oggetto numero 5) è nello stesso luogo dell'avventuriero, occorre tener conto di due casi: trasportata (luogo della sveglia=0) oppure (OR) presente (luogo della sveglia=lu, cioè stesso luogo dell'avventuriero). Al lavoro:

```
3190 rem t2:
3200 if l(5)=0 then print:print "Tic, tac, tic, tac..."
3210 if t2>0 then return
3220 print:print "[[[[[ BOOM! ]]]]]"
3230 if l(5)=0 OR l(5)=lu goto 4000
3240 if l(5)=4 then l(6)=-99:l(7)=-4:l(1)=4
3250 l(8)=-l(5):l(5)=-99:return
```

La linea 3200 stampa "Tic, tac..." solo se l'avventuriero sta trasportando la sveglia (altrimenti è troppo lontano per sentirlo), la 3210 ritorna senza conseguenze se il timer non è a zero, e la 3220 fa scoppiare la bomba quando il timer arriva a zero (senza controllo di luogo, dato che si sente di sicuro in tutto l'appartamento).
Se la sveglia (oggetto 5) è presente (luogo lu) o trasportata (luogo 0), l'agente salta in aria (3230, si potrebbe aggiungere un messaggio), altrimenti prosegue.
Se la sveglia (oggetto 5) si trova in cantina (luogo 4), scompare la cassaforte (oggetto 6) ed appaiono la cassaforte sventrata (oggetto 7, attenzione al solito segno meno) e la chiave (oggetto 1). Si potrebbe anche non far apparire la chiave, e richiedere che il giocatore scriva GUARDA LA CASSAFORTE o, meglio ancora, GUARDA LE MACERIE.
La 3250 fa, in ogni caso, apparire le macerie (oggetto 8) e sparire la sveglia (oggetto 5). Ancora una volta, attenzione al segno meno: dato che le macerie non sono prendibili, devono apparire nel luogo -l(5), non nel luogo l(5). Ho usato l(5) invece di un numero fisso, perchè non è possibile sapere dove esplode la bomba, se non guardando dove si trova la sveglia.

Fate RUN, andate a prendere la sveglia, e scendete in cantina:

Sei in una cantina ammuffita.
Vedo una solida cassaforte.

Cosa devo fare ? carica la sveglia

Cric, cric, cric...

Sei in una cantina ammuffita.
Vedo una solida cassaforte.

Tic, tac, tic, tac...

Cosa devo fare ? lascia la sveglia

Fatto.

Sei in una cantina ammuffita.
Vedo una sveglia di buona marca.
Vedo una solida cassaforte.

Cosa devo fare ? guarda la sveglia

Ticchetta. (meglio andarsene, e di corsa)

Sei in una cantina ammuffita.
Vedo una sveglia di buona marca.
Vedo una solida cassaforte.

Cosa devo fare ? a

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? apri la porta (tanto per passare il tempo)

Non hai la chiave.

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

[[[[ BOOM! ]]]]

Cosa devo fare ? b

Sei in una cantina ammuffita.
Vedo una chiave di sicurezza.
Vedo una cassaforte sventrata.
Vedo un cumulo di macerie.
```

Come dicevo, sarebbe più logico far cercare tra le macerie per trovare la chiave. A proposito, se volete che una certa parola venga ignorata, come il TRA nella frase CERCA TRA LE MACERIE, basta metterla nel dizionario con codice 7.

Ho finito con l'agente segreto ed i suoi microfilm. Il prossimo capitolo mostra alcune comodità del Modulo Base che non ho avuto modo di mostrarvi nel corso di questa avventura.

Capitolo 11: trucchi e comodità varie

Ovvero: come sfruttare il Modulo Base per risparmiare lavoro, come risolvere problemi strani e come aggiungere effetti speciali alle vostre avventure.

Nella costruzione di "Operazione Zanna Bianca", non ho sfruttato a fondo le possibilità offerte dal Modulo Base. Colmo subito la lacuna, illustrandovi varie cose interessanti (e riassumendone alcune già viste di passata nei precedenti capitoli).
Iniziamo dalle subroutine di uso generale. Ne avete già usate due:

- gosub 1500 stampa "Premi <spazio> per continuare" ed attende che il giocatore prema effettivamente la barra spaziatrice. E' utile per evitare che una scritta troppo lunga scorra fuori dal video, o per aggiungere un elemento di suspense alla vicenda.

- gosub 1600 stampa la domanda contenuta in **a$**, ed accetta come risposta soltanto "si" o "no" (o comunque parole che inizino con "s" o "n". Al ritorno, la variabile logica **a** è vera se la risposta era "si", falsa se era "no".

Questa subroutine, invece, non la conoscete ancora:

- gosub 1700 fa una pausa di attesa, con una durata che dipende da **a**. Per la precisione, la pausa dura circa **a** secondi. Esempio:

```
a=5:gosub 1700
```

fa una pausa di 5 secondi. Sono ammessi anche valori decimali e minori di 1. Se la subroutine 1700 viene nuovamente chiamata senza aver cambiato il valore di **a**, ripete lo stesso ritardo dell'ultima volta. Questo vale solo all'interno delle vostre routine di azione, dato che il Modulo Base cambia il contenuto di **a** in molte occasioni.
Una nota: le pause di attesa possono aggiungere suspense e coinvolgere il giocatore, ma possono anche annoiarlo se sono usate troppo spesso o a sproposito. In particolare, non mettete mai pause nell'introduzione, o

comunque quando volete solo dare il tempo di leggere qualcosa. Usate invece il gosub 1500 (premi <spazio> per continuare), che è fatto apposta.

Se trovate comodo aggiungere nuove subroutine di uso generale, che impiegate nelle vostre avventure, potete metterle nello spazio appositamente riservato nel Modulo Base, facendole iniziare alle linee 1800, 1900, 2000... Ricordate di non fare mai un GOTO o GOSUB ad una linea di soli REM, per l'ottimo motivo che vi potrà capitare di togliere i REM per recuperare memoria.

Le variabili

Molte azioni danno un risultato diverso a seconda degli eventi accaduti in precedenza. Ad esempio, CARICA LA SVEGLIA risponde "Già fatto" se la sveglia è già stata caricata, altrimenti la carica. Per sapere se la sveglia è carica, basta guardare se il timer 2 è in funzione. In altri casi, l'informazione necessaria si ricava controllando se un certo oggetto è in un certo luogo (es. Porta/Porta aperta).
Ci sono però situazioni in cui non è possibile sapere se un'azione è stata compiuta o meno, ad esempio se è stata pronunciata una certa parola magica nel tempio di Zot. In questo caso, occorre un posto dove memorizzare l'informazione riguardante l'evento.

Il Modulo Base dispone di dieci variabili apposite, che vengono salvate insieme alla situazione dell'avventura e possono essere liberamente usate per conservare informazioni:

```
v1, v2, v3, v4, v5, v6, v7, v8, v9, v0
```

Una variabile può essere usata, ad esempio, per ricordare lo stato di un'azione complessa, che richiede tre o quattro azioni per essere completata, ad esempio uno SCAVA che va ripetuto quattro o cinque volte:

```
9000 if v3=0 then print "Hai intaccato il terreno.":v3=1:return
9010 if v3=1 then print "Hai fatto una piccola buca.":v3=2:return
9020 if v3=2 then print "La terra e' umida.":v3=3:return
9030 if v3=3 then print "Hai scavato un pozzo.":v3=4:return
9040 print "Non c'e' motivo di scavare ancora.":return
```

Nella linea 9030, sarà anche il caso di far apparire il pozzo scavato. La variabile v3, come dicevo, contiene lo "stato di avanzamento" dei lavori di scavo. Dieci variabili possono sembrare poche, ma sono in realtà sufficienti anche per avventure complesse, dato che molte informazioni sono disponibili osservando il luogo in cui si trovano i vari oggetti.

Load e save

Le routine di registrazione e rilettura della situazione sono piuttosto primitive, ma lo scopo di questo libro è di insegnarvi come scrivere avventure, non come si usa il registratore (o il disk drive, se l'avete). Se vi sentite in grado di farlo, potete modificarle per fare in modo che si possano registrare più situazioni, con nomi o numeri di identificazione differenti.
Attenti però alle trappole: dovete verificare che la situazione richiesta sia effettivamente registrata, e non lasciare che il programma si fermi se non la trova. Dovete tener conto della possibilità che non ci sia più spazio sull'eventuale dischetto, che il nome battuto non sia valido, ecc. Non è difficilissimo, ma non è nemmeno semplice come sembra: come al solito, le rifiniture portano via più lavoro di tutto il resto.

Out of memory

Ahimè, sono certo che prima o poi vi capiterà. Di finire la memoria, intendo. Se scrivete un'avventura di una certa dimensione, finirete con il trovarvi in una situazione molto sgradevole: il programma sta tranquillamente in memoria, ma al RUN si ferma lamentando l'insufficienza della memoria disponibile. Infatti, il programma richiede un certo spazio extra per le variabili numeriche e per i "puntatori" alle variabili stringa (dizionario, descrizioni, e nomi degli oggetti). Se questo spazio manca, non c'è niente da fare.
Beh, non siamo così pessimisti: qualcosa da fare c'è sempre, ad esempio:

- Togliere tutti i REM dal Modulo Base. Ci sono programmi che lo fanno in modo automatico.
- Togliere le routine di azione non usate (difficilmente ne userete 160).
- Togliere le chiamate alle suddette routine di azione, nelle linee intorno

alla 2600, ma solo se avete un'idea ben chiara di quello che state facendo.
- Riunire quante più istruzioni possibile sulla medesima linea di programma.
- Levare tutti gli spazi (meno quelli tra virgolette!).
- Ridurre le dimensioni degli array (linee 1060-1090) allo stretto necessario. (al contrario, potrà capitarvi di doverle aumentare se non bastano per la vostra avventura).

Alla fine di tutto questo, avrete recuperato un bel po' di memoria. Se ne volete dell'altra, avete la scelta tra usare messaggi più corti e trasferire i messaggi in un file su disco, scrivendo una routine che vada a leggerli quando necessario. Oppure potete sfruttare la memoria che il BASIC non può usare, per tenerci i messaggi. Occorre, però, una certa esperienza nell'uso dell'assembly.
Inoltre, eliminando i controlli dalla linea 1170 alla 1220 (cancellando semplicemente le linee) si risparmia ancora un poco di spazio e si riduce l'attesa iniziale. E' però il caso di farlo solo quando il gioco è a posto, altrimenti si rischiano inconvenienti strani se qualche parola o azione non è in ordine (ad esempio: introducendo un'azione, non ne funziona più un'altra).

Effetti speciali

Un'avventura può essere resa più interessante aggiungendo effetti sonori e grafici. Purtroppo, le notevoli capacità del Commodore 64 non sono direttamente utilizzabili da BASIC, se non con complesse, lente e dispendiose sequenze di POKE. Alcune estensioni di BASIC comprendono istruzioni per la grafica e per il suono, ma spesso al prezzo di molta memoria in meno. La soluzione migliore sarebbe quella di routine in assembly, se lo sapete usare. Comunque, qualche effetto sonoro può essere ottenuto a base di POKE senza troppo spreco di memoria.
Per quanto riguarda la grafica, è chiaro che le descrizioni potrebbero essere sostituite da disegni, anche se non è detto che l'avventura ci guadagni: la fantasia del giocatore, se ben stimolata, immaginerà una scena migliore di qualunque disegno. In ogni caso, potete preparare i disegni con uno degli appositi programmi, e richiamarli al momento opportuno (se avete il disk drive).
Alcuni effetti sono però ottenibili anche senza fare uso del disco. In molti

casi, un accorto uso delle possibilità semigrafiche può essere sufficiente per creare qualcosa di inatteso. Un solo suggerimento: potreste disegnare gli oggetti (in risposta a GUARDA IL ...) usando caratteri semigrafici colorati. Il disegno di ciascun oggetto potrebbe anche essere contenuto in una o più stringhe lette dai DATA all'inizio del gioco, ad esempio nell'array do$ (disegni oggetti), insieme alle altre caratteristiche degli oggetti.

Seguite il buon esempio

Se volete imparare un po' di sistemi & trucchetti vari per sfruttare le possibilità offerte dal Modulo Base, studiatevi le due avventure che avete giocato. Troverete spunti sul modo di affrontare problemi che potrebbe capitare anche a voi di dover risolvere. In particolare, ne "L'astronave condannata", meritano attenzione:

- Le azioni PRENDI e LASCIA, con particolare riferimento a casco, tuta, camice e spazio interplanetario (senz'aria).
- L'azione GUARDA INDICATORE.
- Il meccanismo del compartimento stagno.
- Il controllo del reattore.

Anche ne "L'anello di Lucrezia Borgia" potete trovare idee e suggerimenti, ad esempio osservando come funzionano:

- L'orso che insegue (occhio all'azione 1, quella delle direzioni).
- L'effetto dell'elleboro, con la ciotola che si rovescia.
- La sequenza della preghiera, che usa una variabile.
- La battaglia, esempio di uso di semplici effetti speciali.
- La pianura (osservando mappa e azioni).

Non vi do consigli su come trovare queste azioni nel programma: ormai siete esperti.

Conclusione

Ringraziamenti:

- A Chiara Tovena, per l'insostituibile collaborazione nella stesura delle sceneggiature de "L'astronave condannata" e "L'anello di Lucrezia Borgia".

- A Gianni Cavallari, Roberto Cerruti, Marco Morocutti, Lucia Bonazzi, Luisa Moleri, Fulvio Francesconi, Manuela Spotti, ed a vari amici di passaggio, per i collaudi de "L'astronave condannata" e "L'anello di Lucrezia Borgia".

- A Luisa Moleri e Fulvio Francesconi de "Il Computer" di Brescia, per avermi messo a disposizione attrezzature varie, e per aver amichevolmente sopportato i miei grovigli seriali (RS-232) nel loro simpatico computer shop.

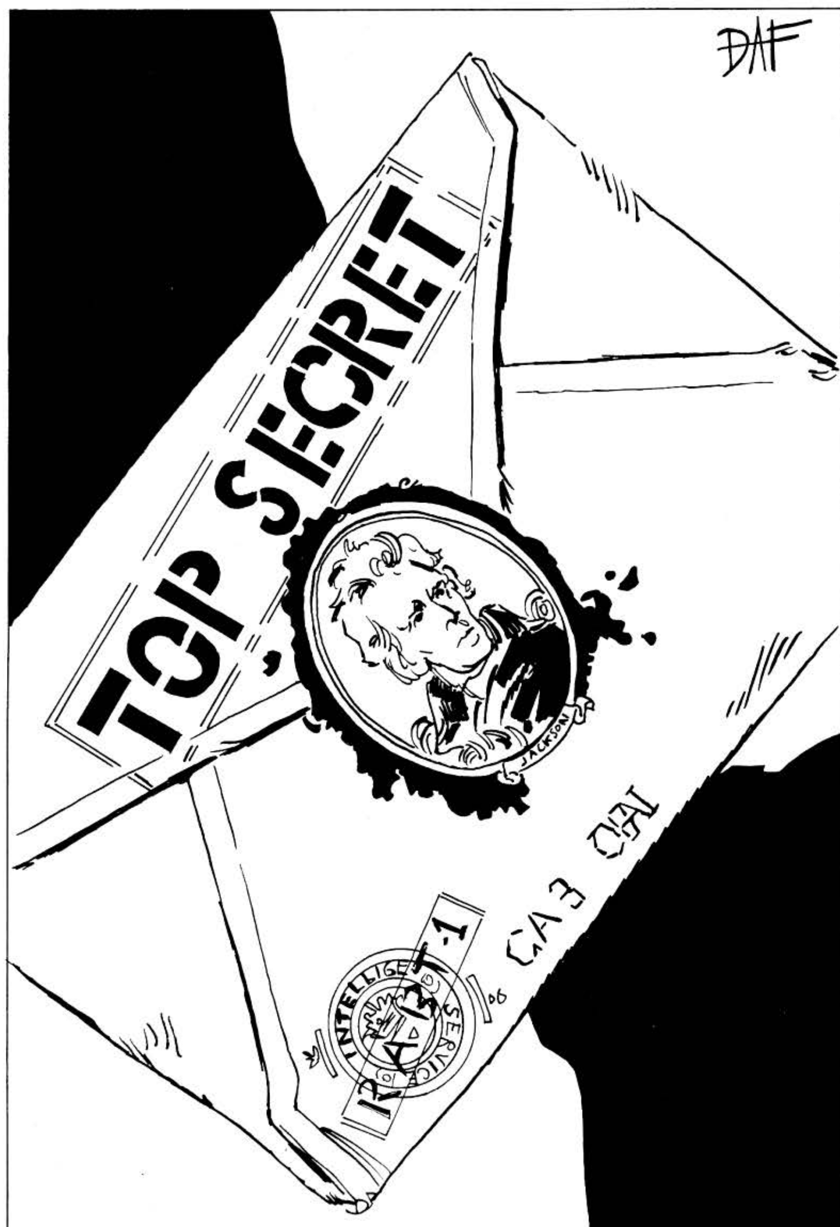
- Al mio Macintosh, che non mi ha mai tradito, nonostante gli improperi ricevuti per la lentezza a volte esasperante.

- A Roberto Pancaldi, ed a tutta la Divisione Libri Jackson, per l'assistenza e la collaborazione organizzativa a tempo di record.

A tutti, infine:

Buona avventura!

Enrico Colombini


DAF
TOP SECRET
JACKSON

Appendice A: listato del Modulo Base

```
100 rem ** avventura:  interprete  e modulo  base **
110 rem **          di enrico colombini          **
120 rem ** (c)1985 dinosoft  e jackson editrice  **
130 rem
140 goto710:rem  main
150 rem
160 rem - cerca p$ in dizionario, c=codice (0 se assente) -
170 rem
180 i=1:f=fd
190 a=int((i+f)/2):a$=dz$(a):ifp$=a$thenc=dz(a):goto240
200 ifp$>a$theni=a+1
210 ifp$<a$thenf=a-1
220 ifi<=fthen190
230 c=0
240 return
250 rem
260 rem - estrae p$ da in$[in], trova codice c, salta articoli -
270 rem
280 c=0
290 c$=mid$(in$,in,1):ifc$=" "orc$="'"thenin=in+1:goto290
300 ifin>lithenp$="":goto360
310 a=in:rem  inizio
320 c$=mid$(in$,in,1)
330 ifc$<>" "andc$<>"'"andin<=lithenin=in+1:goto320
340 p$=mid$(in$,a,in-a):gosub180:rem   cerca
350 ifc=7then280:rem  articolo
360 return
370 rem
380 rem - cerca azione a, a=azione (0 se non trovata) -
390 rem
400 i=1:f=fa:n=a
410 a=int((i+f)/2):m=ca(a):ifn=mthena=az%(a):goto460
420 ifn>mtheni=a+1
430 ifn<mthenf=a-1
440 ifi<=fthen410
450 a=0
```

```
460 return
470 rem
480 rem - esegue azione a, a=0 se non trovata -
490 rem
500 gosub400:ifa=0then550
510 ifc2=0ora>0then530:rem  solo verbo o no test
520 a=-a:ifog=0thenprint"-  Qui non ne vedo.":goto540
530 gosub2500:rem  esegue
540 a=1
550 return
560 rem
570 rem - elenca oggetti in luogo l, con prefisso p$, a=0 se
    nessuno -
580 rem
590 poke k,k3:a=0:fori=1tofo
600 ifabs(l(i))=lthenprintp$;og$(i);".":a=1
610 next:poke k,k0:return
620 rem
630 rem - og=indice oggetto c2, 0 se non pres. o trasp. -
640 rem
650 og=0:fori=1tofo
660 ifco%(i)=c2thenifabs(l(i))=luorl(i)=0thenog=i:i=fo
670 next:return
680 rem
690 rem - main (parser) -
700 rem
710 gosub1060:rem  init
720 gosub4500:rem  introduzione
730 gosub1500:rem  <sp>
740 rem
750 rem ciclo di gioco:
760 poke k,k2:print:print"Sei  ";de$(lu);".":poke  k,k0:rem
    descrizione
770 l=lu:p$="Vedo  ":gosub590:rem  oggetti
780 gosub970:rem  tempo
790 print:print
800 in$="":poke  k,k1:input"Cosa  devo fare  ";in$:ifin$=""then800
810 poke k,k0:print:li=len(in$):in=1:gosub280:p1$=p$:c1=c
820 ifp1$=""thenprint"-  Beh ?":goto760
```

```
830 ifright$(p1$,2)="re"thenprint"-   Dammi  del  tu,  per
    favore.":goto760
840 ifc1=0andp1$<>""thenprint"-   Non  conosco  il  verbo  '"
    p1$"'.":goto760
850 gosub280:p2$=p$:c2=c:ifc2>0andc2<7then850:rem    no articoli
860 ifc2=0andp2$<>""thenprint"-   Non  conosco  la  parola  '"
    p2$"'.":goto760
870 ifc2<>0thengosub650:rem   og=indice
880 n1=lu*10000:n2=c1*100
890 a=n1+n2+c2:gosub500:ifagoto760:rem   verbo+nome  in  lu
900 ifc2<>0thena=n1+n2+99:gosub500:ifagoto760:rem    verbo+x  in  lu
910 a=n2+c2:gosub500:ifagoto760:rem   verbo+nome  generico
920 ifc2<>0thena=n2+99:gosub500:ifagoto760:rem    verbo+x  generico
930 print"-  Non  capisco.":goto760
940 rem
950 rem  -  tempo  -
960 rem
970 ift1thent1=t1-1:gosub3000
980 ift2thent2=t2-1:gosub3200
990 ift3thent3=t3-1:gosub3400
1000 ift4thent4=t4-1:gosub3600
1010 ift5thent5=t5-1:gosub3800
1020 return
1030 rem
1040 rem  -  init  -
1050 rem
1060 dimdz$(150),dz(150):rem   dizionario
1070 dimde$(98),di$(98):rem   mappa
1080 dimca(160),az%(160):rem   azioni
1090 dimog$(50),co%(50),l(50):rem   oggetti
1095 di=0:rem  0=tape,1=disk
1100 f$="situazione":rem   file situazione
1110 rem  fd,fm,fa,fo=0
1115 print  chr$(14)chr$(8):rem   lock minuscole
1120 print:print"Un  attimo  di  pazienza...";
1130 reada$:ifa$<>"fd"thenfd=fd+1:dz$(fd)=a$:readdz(fd):goto1130
1140 reada$:ifa$<>"fm"thenfm=fm+1:de$(fm)=a$:readdi$(fm):goto1140
1150 reada$:ifa$<>"fa"thenfa=fa+1:ca(fa)=val(a$):readaz%(fa):
    goto1150
```

```
1160 reada$:ifa$<>"fo"thenfo=fo+1:og$(fo)=a$:readco%(fo),l(fo):
    goto1160
1170 fori=2tofd:rem  --- controllo  dizionario  (eliminabile)  ---
1180 ifdz$(i)<=dz$(i-1)thenprint"Errore:   "dz$(i)"  <=  "dz$(i-1):end
1190 next
1200 fori=2tofa:rem  --- controllo  azioni  (eliminabile)  ---
1210 ifca(i)<=ca(i-1)thenprint"Errore:   "ca(i)"  <=  "ca(i-1):end
1220 next
1230 poke 53280,9:poke  53281,11:rem  sfondo
1240 k=646:k0=7:k1=15:k2=13:k3=8:poke   k,k0:rem  colori
1250 print:print:return
1470 rem
1480 rem  -  <sp>  per  continuare  -
1490 rem
1500 print:print"-  Premi  <spazio>  per  continuare  -";
1510 geta$:ifa$<>"  "then1510
1520 print:print:return
1570 rem
1580 rem  -  domanda  a$,  a=vero  se  si'  -
1590 rem
1600 printa$;:b$="":input"   ";b$:b$=left$(b$,1)
1610 ifb$<>"s"andb$<>"n"then1600
1620 a=(b$="s"):return
1670 rem
1680 rem  -  pausa  a  secondi  circa  -
1690 rem
1700 fori=1to1000*a:next:return
1710 rem
1720 rem  *  (mettere  qui  altre  subroutine  di  uso  generale)  *
2470 rem
2480 rem  -  esegue  azione  a  -
2490 rem
2500 onint((a-1)/20)goto2510,2530,2550,2570,2590,2610,2630,2650
2510 onagoto5000,5200,5400,5600,5800,6000,6200,6400,6600,6800
2520 ona-10goto7000,7200,7400,7600,7800,8000,
    8200,8400,8600,8800
2530 ona-20goto9000,9200,9400,9600,9800,
    10000,10200,10400,10600,10800
2540 ona-30goto11000,11200,11400,11600,11800,
```

```
     12000,12200,12400,12600,12800
2550 ona-40goto13000,13200,13400,13600,13800,
     14000,14200,14400,14600,14800
2560 ona-50goto15000,15200,15400,15600,15800,
     16000,16200,16400,16600,16800
2570 ona-60goto17000,17200,17400,17600,17800,
     18000,18200,18400,18600,18800
2580 ona-70goto19000,19200,19400,19600,19800,
     20000,20200,20400,20600,20800
2590 ona-80goto21000,21200,21400,21600,21800,
     22000,22200,22400,22600,22800
2600 ona-90goto23000,23200,23400,23600,23800,
     24000,24200,24400,24600,24800
2610 ona-100goto25000,25200,25400,25600,25800,
     26000,26200,26400,26600,26800
2620 ona-110goto27000,27200,27400,27600,27800,
     28000,28200,28400,28600,28800
2630 ona-120goto29000,29200,29400,29600,29800,
     30000,30200,30400,30600,30800
2640 ona-130goto31000,31200,31400,31600,31800,
     32000,32200,32400,32600,32800
2650 ona-140goto33000,33200,33400,33600,33800,
     34000,34200,34400,34600,34800
2660 ona-150goto35000,35200,35400,35600,35800,
     36000,36200,36400,36600,36800
2670 printa" ???":return:rem  per sicurezza
2920 rem
2930 rem
2940 rem *** fine interprete,  inizio avventura ***
2950 rem
2960 rem
2970 rem - timer -
2980 rem
2990 rem t1:
3000 print:print"T1":return
3190 rem t2:
3200 print:print"T2":return
3390 rem t3:
3400 print:print"T3":return
```

```
3590 rem t4:
3600 print:print"T4":return
3790 rem t5:
3800 print:print"T5":return
3970 rem
3980 rem - morto -
3990 rem
4000 gosub 1500:poke 53280,2:rem  <sp>
4010 rem --- mettere qui il finale negativo ---
4450 a$="Vuoi giocare ancora":gosub1600:if(a)thenrun:rem   riparte
4460 print:print:print"Ciao!":print:end
4470 rem
4480 rem - introduzione  -
4490 rem
4500 print chr$(147):print:print:print
4510 rem --- mettere qui introduzione avventura ---
4520 rem --- ricordarsi  lu=luogo iniziale ---
4950 lu=1:print:return
4960 rem
4970 rem - azioni: -
4980 rem
4990 rem 1: (direzioni)
5000 a=val(mid$(di$(lu),2*c1-1,2))
5010 ifa=0thenprint"-  Di li' non puoi andare.":return
5020 lu=a:return
5190 rem 2: (prendi)
5200 ifl(og)=0thenprint"-  Gia' fatto.":return
5210 ifl(og)<0thenprint"-  Non e' possibile.":return
5220 l(og)=0:print"Fatto.":return
5390 rem 3: (lascia)
5400 ifog=0orl(og)<>0thenprint"-  Non ce l'hai.":return
5410 l(og)=lu:print"Fatto.":return
5590 rem 4: (guarda)
5600 print"Non noto nulla di particolare.":return
5790 rem 5: (save)
5800 if di=0 then open 3,1,1,f$:goto 5820:rem tape
5810 open 15,8,15,"i":open 3,8,3,"@0:"+f$+",s,w":rem  disk
5820 fori=1tofo:print#3,l(i):next
5830 print#3,lu:print#3,t1:print#3,t2:
```

```
     print#3,t3:print#3,t4:print#3,t5
5840 print#3,v1:print#3,v2:print#3,v3:print#3,v4:print#3,v5
5850 print#3,v6:print#3,v7:print#3,v8:print#3,v9:print#3,v0
5860 close 3:close 15:return
5990 rem 6: (load)
6000 if di=0 then open 3,1,0,f$:goto6020:rem  tape
6010 open 15,8,15,"i":open  3,8,3,f$+",s,r":rem  disk
6020 fori=1tofo:input#3,l(i):next
6030 input#3,lu,t1,t2,t3,t4,t5,v1,v2,v3,v4,v5,v6,v7,v8,v9,v0
6040 close 3:close 15:return
6190 rem 7: (cosa)
6200 print"Possiedi:":print:l=0:p$="-    ":gosub590
6210 ifa=0thenprint"-  Un bel nulla."
6220 return
6390 rem 8:
6400 print8:return
6590 rem 9:
6600 print9:return
6790 rem 10:
6800 print10:return
....
```

(continua così, con le routine di azione a passo 200, fino alla 35400)

```
.....
35590 rem 154:
35600 print154:return
35790 rem 155:
35800 print155:return
35990 rem 156:
36000 print156:return
36190 rem 157:
36200 print157:return
36390 rem 158:
36400 print158:return
36590 rem 159:
36600 print159:return
36790 rem 160:
36800 print160:return
```

```
39970 rem
39980 rem - dizionario:
39990 rem
40000 data "?",13,"a",5,"agli",7,"al",7,"all",7
40010 data "alla",7,"alle",7,"allo",7,"alto",5
40020 data "b",6,"basso",6
40030 data "col",7,"con",7,"cosa",13
40040 data "e",3,"est",3
40050 data "gli",7,"guarda",10
40060 data "i",7,"il",7,"inventario",13
40070 data "l",7,"la",7,"lascia",9
40080 data "le",7,"lo",7,"load",12
40090 data "n",1,"nord",1
40100 data "o",4,"ovest",4
40110 data "posa",9,"prendi",8
40120 data "riprendi",12
40130 data "s",2,"sali",5,"salva",11
40140 data "save",11,"scendi",6,"sud",2
40150 data "un",7,"una",7,"uno",7
40160 data "w",4
40170 data "fd": rem --- ricordarsi l'ordine alfabetico!!! ---
44970 rem
44980 rem - mappa:
44990 rem
45000 data "fm": rem --- luoghi e mappa, dal luogo n.1 ---
49970 rem
49980 rem - azioni:
49990 rem
50000 data 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 data 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 data "fa": rem --- ricordarsi l'ordine numerico!!! ---
54970 rem
54980 rem - oggetti:
54990 rem
55000 data "fo": rem --- descrizioni e dati oggetti, dal n.1 ---
```

Appendice B: uso del Modulo Base: riassunto

Mappa:

Inserire descrizioni luoghi e direzioni come data dalla linea 45000 in poi, in ordine di numero di luogo (dal n.1) e terminando con "fm".
Le direzioni si indicano con 12 cifre: le prime due indicano il numero del luogo adiacente a Nord, le successive quello a Sud, poi ad Est, Ovest, Alto, Basso. Se lo spostamento non è possibile, mettere 00 (doppio zero) come numero del luogo. Esempio:

```
45000 data "in fondo al pozzo","000000002400"
```

Si può andare solo in alto, e si finisce nel luogo 24 (il 24º dell'elenco).

Dizionario:

Inserire parole e codici come data dalla linea 40000 in poi, ricordandosi di mantenere l'ordine alfabetico e terminare con "fd". Esempio:

```
40000 data "?",13,"a",5,"accendi",35,"agli",7
```

La nuova parola ACCENDI, con codice 35, è stata inserita in mezzo a quelle già presenti nel dizionario, in ordine alfabetico e minuscola.

Oggetti:

Inserire descrizione, codice parola di riferimento e luogo iniziale come data a partire dalla linea 55000, in ordine di numero di oggetto (dal n.1) e terminando con "fo". Esempio:

```
55000 data "un libro Jackson",42,13
```

La parola a cui l'oggetto risponde (probabilmente LIBRO) si trova nel dizionario con codice 42. Il libro si trova, all'inizio del gioco, nel luogo 13.

Se un oggetto non è prendibile, va indicato con codice negativo:

```
55010 data "un masso di granito rosa",44,-13
```

Se l'oggetto non è in gioco, si indica -99 (il Limbo) come numero di luogo:

```
55020 data "un brontosauro sonnolento",52,-99
```

Sel'oggetto si deve trovare, all'inizio del gioco, in possesso dell'avventuriero, si indica il luogo zero:

```
55030 data "un orologio scarico",55,0
```

Il numero di un oggetto corrisponde alla sua posizione nell'elenco dei data. Riguardando gli esempi fatti, il libro sarebbe l'oggetto 1, il masso l'oggetto 2, il brontosauro l'oggetto 3, e l'orologio l'oggetto 4.

Azioni:

Inserire codice frase e numero azione come data dalla linea 50000 in poi, ricordandosi di mantenere l'ordine numerico e terminare con "fa".
Il codice di frase è un numero composto da:

- Numero del luogo (2 cifre).
- Codice della prima parola (2 cifre).
- Codice della seconda parola (2 cifre).

Gli zeri iniziali possono poi essere eliminati.

Con frasi di una sola parola, per il codice di frase sono possibili queste due combinazioni (in ordine di precedenza):

PRENDI (nel luogo 13)	131000	(luogo 13/PRENDI/nessuna parola)
PRENDI (dovunque)	001000	(dovunque/PRENDI/nessuna parola)

Con frasi di due parole (articoli e preposizioni non contano), ci sono quattro possibili combinazioni, sempre in ordine di precedenza:

"PRENDI LIBRO"	(luogo 13)	131042	(luogo 13/PRENDI/LIBRO)
"PRENDI X"	(luogo 13)	131099	(luogo 13/PRENDI/qualunque parola)
"PRENDI LIBRO"	(dovunque)	001042	(dovunque/PRENDI/LIBRO)
"PRENDI X"	(dovunque)	001099	(dovunque/PRENDI/qualunque parola)

Al codice di frase segue il numero dell'azione da eseguire. Se questo numero è negativo, l'azione è eseguita soltanto se la seconda parola della frase indica un oggetto presente o trasportato. Esempio:

```
50010 data 1000,4,1042,-12,1099,-4,1100,5,1200,6,1300,7
```

Il nuovo codice di frase 1042, è stato inserito in mezzo a quelli già presenti nella tavola, in ordine numerico, seguito dal numero dell'azione da eseguire (-12). Se 10 è il codice di GUARDA nel dizionario, e 42 è il codice di LIBRO, la frase GUARDA IL LIBRO fa eseguire la subroutine di azione numero 12 ma solo se l'oggetto citato (il libro) è presente o trasportato (a causa del segno meno davanti al 12).

Subroutine di azione:

La subroutine di azione numero 1 inizia alla linea 5000, la numero 2 alla 5200, e così via di 200 in 200 fino alla subroutine numero 160, che si trova alla linea 36800. I numeri di linea non vanno modificati.
Ogni subroutine di azione è composta di normali istruzioni BASIC e deve terminare con l'istruzione RETURN.
Le informazioni sull'avventura sono contenute in alcune variabili fondamentali:

lu	Numero del luogo corrente.
og	Numero dell'oggetto citato nella frase (0 se non è un oggetto).
l(n)	Luogo dell'oggetto numero n (0=trasp., -99=Limbo), negativo se l'oggetto non è prendibile.
v0...v9	Variabili libere, usabili a piacere.
t1...t5	Timer, partono mettendoci un numero maggiore di zero.

Esempio di azione numero 12 (linea 7200):

```
7200 if l(1) <> 0 then print "Non hai in mano il libro." : return
7210 print "Leggi la parola 'FUOCO', poi il libro scompare."
7220 l(1) = -99 : return : rem mette il libro nel limbo
```

L'oggetto numero 1 (il libro) è quello elencato per primo nei data alla linea 55000. l(1) è dunque il luogo in cui si trova il libro. Occorre fare attenzione agli oggetti non prendibili, il cui numero di luogo è e deve restare negativo (a meno che non si voglia renderli prendibili).

Indirizzi utili (numeri di linea):

gosub 1500	- Premi <spazio> per continuare:
gosub 1600	Stampa la domanda contenuta in a$, torna con **a**=vero se la risposta è "sì", falso se è "no".
gosub 1700	Aspetta **a** secondi circa.
gosub 3000	Subroutine di timer 1 (eseguita automaticamente ad ogni turno se il timer 1 è attivo). Le successive sono alle linee 3200 (t2), 3400 (t3), 3600 (t4) e 3800 (t5)
gosub 5000	Subroutine di azione 1 (eseguita automaticamente), le successive sono alla 5200 (2), 5400 (3), ecc.

Per chi vuole approfondire:

Il Quaderno Jackson "Scrivere un'avventura" descrive in dettaglio la teoria ed il funzionamento dell'interprete. Il programma è leggermente diverso, ma il principio di funzionamento è il medesimo.
Nella pagina seguente, sono riportate le principali variabili del programma, usabili nelle routine di azione.

Dizionario:

fd	|	Numero vocaboli in dizionario.
dz$(1..fd)	|	Vocaboli, in ordine alfabetico.
dz%(1..fd)	|	Codici dei vocaboli corrispondenti in dz$().

Mappa:

fm	|	Numero luoghi in mappa.
de$(1..fm)	|	Descrizioni luoghi, in ordine di numero.
di$(1..fm)	|	Collegamenti del luogo descritto nel corrispondente
	|	elemento di de$() con i sei luoghi adiacenti (N/S/E/O/A/B).

Azioni:

fa	|	Numero azioni nella tavola.
ca(1..fa)	|	Codici di azione, in ordine numerico.
az%(1..fa)	|	Numero azione corrispondente da eseguire, negativo
	|	se è richiesta la presenza dell'oggetto citato.

Oggetti:

fo	|	Numero oggetti presenti.
og$(1..fo)	|	Descrizioni oggetti, in ordine di numero oggetto.
co%(1..fo)	|	Codice nel dizionario dell'oggetto descritto nel
	|	corrispondente elemento di og$().
l(1..fo)	|	Luogo dell'oggetto corrispondente:
	|	1..fm luogo indicato, prendibile.
	|	-1..-fm luogo indicato, non prendibile.
	|	0 trasportato dall'avventuriero.
	|	-99 limbo (fuori gioco).

Varie:

lu	|	Luogo corrente dell'avventuriero.
og	|	Numero oggetto citato come seconda parola,
	|	zero se non presente o trasportato.
p1$,p2$	|	Prima e seconda parola valida della frase.
c1,c2	|	Codici nel dizionario di p1$ e p2$

Scritto dal più noto autore italiano di giochi di avventura, questo libro insegna come scrivere avventure sul proprio computer.
Il "Modulo Base", incluso nella cassetta consente anche ai principianti del BASIC di scrivere un vero gioco di avventura di media complessità, con il minimo lavoro per l'autore ed il massimo divertimento per i giocatori. I programmatori esperti vi troveranno una miniera di idee e soluzioni.

Contiene:

- Due avventure pronte da giocare: "L'astronave condannata" (tempo medio di gioco: 2 ore) e "L'anello di Lucrezia Borgia" (molto, molto di più...).
- Il "Modulo Base", programma per la scrittura di avventure.
- Una raccolta di fogli per il disegno di mappe.
- Un corso completo per la creazione di giochi di avventura.
- Il listato commentato del "Modulo Base".
- Tabelle riassuntive per gli autori di avventure.

L. 20.000

COD. CC260 ISBN 88-7056-339-1